Efficiency of Ontology Mapping Approaches

Marc Ehrig and Steffen Staab

Institute AIFB, University of Karlsruhe

Abstract. (Semi-)automatic mapping — also called (semi-)automatic alignment — of ontologies is a core task to achieve interoperability when two agents or services use different ontologies. In the existing literature, the focus has so far been on improving the quality of mapping results. We here consider QOM, Quick Ontology Mapping, as a way to trade off between effectiveness (i.e. quality) and efficiency of the mapping generation algorithms. We show that QOM has lower run-time complexity than existing prominent approaches. Then, we show in experiements that this theoretical investigation translates into practical benefits. While QOM gives up some of the possibilities for producing high-quality results in favor of efficiency, our experiments show that this loss of quality is marginal.

1 Introduction

Semantic mapping[1] between ontologies is a necessary precondition to establish interoperation between agents or services using different ontologies. In recent years we have seen a range of research work on methods proposing such mappings [1–3]. The focus of the previous work, however, has been laid exclusively on improving the *effectiveness* of the approach (i.e. the quality of proposed mappings such as evaluated agains some human judgement given either a posteriori or a priori).

When we tried to apply these methods to some of the real-world scenarios we address in other research contributions (e.g., [4]), we found that existing mapping methods were not suitable for the ontology integration task at hand, as they all neglected *efficiency*. To illustrate our requirements: We have been working in realms where light-weight ontologies are applied such as the ACM Topic hierarchy with its 10^4 concepts or folder structures of individual computers, which corresponded to 10^4 to 10^5 concepts. Finally, we are working with Wordnet exploiting its 10^6 concepts (cf. [5]). When mapping between such light-weight ontologies, the trade-off that one has to face is between effectiveness and efficiency. For instance, consider the knowledge management platform built on a Semantic Web And Peer-to-peer basis in SWAP [4]. It is not sufficient to provide its user with the best possible mapping, it is also necessary to answer his queries within a few seconds — even if two peers use two different ontologies and have never encountered each other before.

In this paper we present an approach that considers both the quality of mapping results as well as the run-time complexity. Our hypothesis is that mapping algorithms may be streamlined such that the loss of quality (compared to a standard baseline) is

[1] Frequently also called alignment.

marginal, but the improvement of efficiency is so tremendous that it allows for the ad-hoc mapping of large-size, light-weight ontologies. To substantiate the hypothesis, we outline a comparison of the worst-case run-time behavior (given in full detail in [6]) and we report on a number of practical experiments. The approaches used for our (unavoidably preliminary) comparison represent a wide range of different classes of algorithms for ontology mapping. From these approaches we can already infer a good performance of our new efficient approach QOM, for which complexity is of $O(n)$ (measuring with n being the number of the entities in the ontologies) against $O(n^2)$ for the approach that comes closest.

The remainder of the paper starts with a clarification of terminology (Section 2). To compare the worst-case run-time behavior of different approaches, we then describe a canonical process for ontology mapping that subsumes the different approaches compared in this paper (Section 3). The process is a core building block for later deriving the run-time complexity of the different mapping algorithms. Section 4 presents our toolbox to analyse these algorithms. In Section 5, different approaches for proposing mappings are described and aligned to the canonical process. The way to derive their run-time complexity is outlined in Section 6. Experimental results (Section 7) complement the comparison of run-time complexities.

2 Terminology

2.1 Ontology

As we currently focus on light-weight ontologies, we build on RDF/S[2] to represent them. To facilitate the further description, we briefly summarize its major primitives and introduce some shorthand notations. An RDF model is described by a set of statements, each consisting of a subject, a predicate and an object. An ontology O is defined by its set of Concepts $\mathcal{C}$ (instances of "rdf:Class") with a corresponding subsumption hierarchy H_C (a binary relation corresponding to "rdfs:subClassOf"). Relations $\mathcal{R}$ (instances of "rdf:Property") exist between single concepts. Relations are arranged alike in a hierarchy H_R ("rdfs:subPropertyOf"). An entity $i \in \mathcal{I}$ may be an instance of a class $c \in \mathcal{C}$ ("rdf:type"). An instance $i \in \mathcal{I}$ may have one j or many role fillers from $\mathcal{I}$ for a relation r from $\mathcal{R}$. We also call this type of triple (i, r, j) a property instance.

2.2 Mapping

We here define our use of the term "mapping". Given two ontologies O_1 and O_2, mapping one ontology onto another means that for each entity (concept C, relation R, or instance I) in ontology O_1, we try to find a corresponding entity, which has the same intended meaning, in ontology O_2.

Definition 1. *We define an ontology mapping function,* map*, based on the vocabulary,* $\mathcal{E}$*, of all terms* $e \in \mathcal{E}$ *and based on the set of possible ontologies,* $\mathcal{O}$ *as a partial function:*

- $\mathrm{map} : \mathcal{E} \times \mathcal{O} \times \mathcal{O} \rightharpoonup \mathcal{E}$*, with*

[2] http://www.w3.org/RDFS/

– $\forall e \in O_1(\exists f \in O_2 : \text{map}(e, O_1, O_2) = f \vee \text{map}(e, O_1, O_2) = \perp)$
indicating that an entity is mapped to exactly one other entity or none.

A term e interpreted in an ontology O is either a concept, a relation or an instance, i.e. $e_{|O} \in \mathcal{C} \cup \mathcal{R} \cup \mathcal{I}$. We usually write e instead of $e_{|O}$ when the ontology O is clear from the context of the writing. We write $\text{map}_{O_1,O_2}(e)$ for $\text{map}(e, O_1, O_2)$. We derive a relation map_{O_1,O_2} by defining $\text{map}_{O_1,O_2}(e, f) \Leftrightarrow \text{map}_{O_1,O_2}(e) = f$. We leave out O_1, O_2 when they are evident from the context and write $\text{map}(e) = f$ and $\text{map}(e, f)$, respectively. Once a (partial) mapping, map, between two ontologies O_1 and O_2 is established, we also say "*entity e is mapped onto entity f*" iff $\text{map}(e, f)$. A pair of entities (e, f) that is not yet in map and for which appropriate mapping criteria still need to be tested is called a *candidate mapping*.

2.3 Example

The following example illustrates an example mapping. Two ontologies O_1 and O_2 describing the domain of car retailing are given (Figure 1). A reasonable mapping between the two ontologies is given in Table 1 as well as by the dashed lines in the figure.

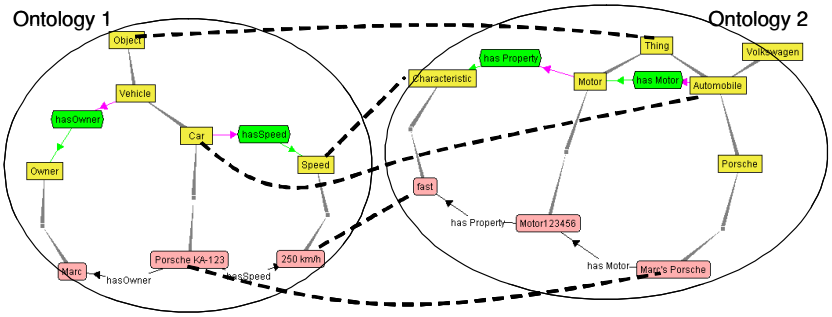

Fig. 1. Example Ontologies and their Mappings

Ontology O_1	Ontology O_2
Object	Thing
Car	Automobile
Porsche KA-123	Marc's Porsche
Speed	Characteristic
250 km/h	fast

Table 1. Mapping Table for Relation $\text{map}_{O_1,O_2}(e, f)$

3 Process

We briefly introduce a canonical process that subsumes all the mapping approaches we are aware of.[3] Figure 2 illustrates its six main steps. It is started with two ontologies, which are going to be mapped onto one another, as its input:

1. Feature engineering transforms the initial representation of ontologies into a format digestible for the similarity calculations. For instance, the subsequent mapping process may only work on a subset of RDFS primitives.

2. Selection of Next Search Steps. The derivation of ontology mappings takes place in a search space of candidate mappings. This step may choose, e.g., to compute the similarity of a subset of candidate concepts pairs $\{(c_1, c_2)|c_1 \in O_1, c_2 \in O_2\}$ and to ignore others.

3. Similarity Computation determines similarity values between pairs of entities e, f based on their definitions in O_1 and O_2, respectively.

4. Similarity Aggregation. In general, there may be several similarity values for a candidate pair of entities e, f from two ontologies O_1, O_2, e.g. one for the similarity of their labels and one for the similarity of their relationship to other terms. These different similarity values for one candidate pair must be aggregated into a single aggregated similarity value.

5. Interpretation uses the individual or aggregated similarity values to derive mappings between entities from O_1 and O_2. Some mechanisms here are, e.g., to use thresholds for similarity mappings, to perform relaxation labelling, or to combine structural and similarity criteria.

6. Iteration. Several algorithms perform an iteration over the whole process in order to bootstrap the amount of structural knowledge. Iteration may stop when no new mappings are proposed. Note that in a subsequent iteration one or several of steps 1 through 5 may be skipped, e.g. because all features might already be available in the appropriate format or because some similarity computation might only be required in the first round.

Eventually, the output returned is a mapping table representing the relation map_{O_1,O_2}.

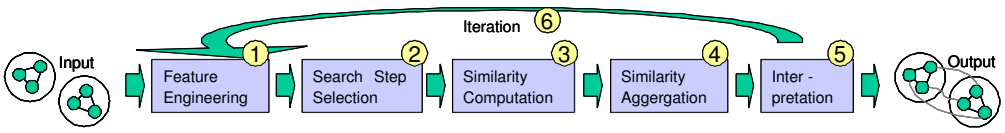

Fig. 2. Mapping Process

[3] The process is inspired by CRISP-DM, http://www.crisp-dm.org/, the CRoss Industry Standard Process for Data Mining.

4 A Toolbox of Data Structures and Methods

The principal idea of this section is to provide a toolbox of data structures and methods common to many approaches that determine mappings. This gives us a least common denominator based on which concrete approaches instantiating the process depicted in Figure 2 can be compared more easily.

Considering a candidate mapping (e, f) requires the investigation of these entities and their definitions in O_1 and O_2. The process in Section 3 distinguishes the investigation in two steps. In the first step, described in the next subsection, features for the two entities e, f are defined to describe them in a concise way. Then we apply similarity measures to the features to actually compare the two entities along different dimensions.

4.1 Features

Features typically extracted are derived from intensional as well as from extensional ontology definitions:

- *Identifiers*: i.e. strings with dedicated formats, such as unified resource identifiers (URIS) or RDF labels.
- *RDF/S Primitives*: such as properties or subclass relations
- *Derived Features*: which constrain or extend simple RDFS primitives (most-specific-class-of-instance)
- *Aggregated Features*: i.e. more than one simple RDFS primitive, e.g. a sibling is every instance-of the parent-concept of an instance
- *OWL Primitives*: such as the sameAs relations
- *Domain Specific Features*: i.e. features which only apply to a certain domain with a predefined shared ontology, e.g. in an application where files are represented as instances and the relation hashcode-of-file is defined, we use this feature to compare representations of concrete files

4.2 Similarity Measures

Definition 2. *We define a similarity measure for comparison of ontology entities as a function as follows (cf. [7]):*

- $sim : \mathcal{E} \times \mathcal{E} \times \mathcal{O} \times \mathcal{O} \rightharpoonup [0, 1]$

Different similarity measures $sim_k(e, f, O_1, O_2)$ are indexed through a label k.Further, we leave out O_1, O_2 when they are evident from the context and write $sim_k(e, f)$. The following similarity measures are needed to compare the previously defined ontology features.

- *Object Equality* is based on existing logical assertions — especially assertions from previous iterations: $sim_{obj}(a, b) := \{1|map_{prev}(a) = b, 0|otherwise\}$
- *Explicit Equality* checks whether a logical assertion already forces two entities to be equal: $sim_{exp}(a, b) := \{1|\exists statement(a, \text{“}sameAs'', b), 0|otherwise\}$

– *String Similarity* measures the similarity of two strings on a scale from 0 to 1 (cf. [8]) based on Levenshtein's edit distance, ed [9].
$sim_{str}(c,d) := max(0, \frac{min(|c|,|d|)-ed(c,d)}{min(|c|,|d|)})$
– *SimSet*: For many features we have to determine to what extent two sets of entities are similar. To remedy the problem multidimensional scaling [10] measures how far two entities are from all other entities and assumes that if they have very similar distances to all other entities, they must be very similar:
$sim_{set}(E,F) = \frac{\sum_{e \in E} \boldsymbol{e}}{|E|} \cdot \frac{\sum_{f \in F} \boldsymbol{f}}{|F|}$
with $\boldsymbol{e} = (sim(e,e_1), sim(e,e_2), \ldots, sim(e,f_1), sim(e,f_2), \ldots)$, $\boldsymbol{f}$ analogously
– *Aggregated Similarity:* $sim_{agg}(e,f) = \frac{\sum_{k=1\ldots n} w_k \cdot adj(sim_k(e,f))}{\sum_{k=1\ldots n} w_k}$
with w_k being the weight for each individual similarity measure, and adj being a function to transform the original similarity value ($adj : [0,1] \rightarrow [0,1]$) e.g. a sigmoid function.

4.3 Interpretation

To derive a mapping from the similarity values we interpret similarity as follows:

– $(sim(e,f) > t) \rightarrow (map(e) = f)$, with $t \in [0 \ldots 1]$ being a defined threshold

5 Approaches to Determine Mappings

In the following we now use the toolbox, and extend it, too, in order to define a range of different mapping generation approaches. In the course of this section we present our novel Quick Ontology Mapping approach - QOM.

5.1 NOM - Naive Ontology Mapping

Our Naive Ontology Mapping (NOM) constitutes a straight forward baseline for later comparisons. It is defined by the steps of the process model as follows:

1. Feature Engineering Firstly, NOM uses RDF triples as features.

2. Search Step Selection All entities of the first ontology are compared with all entities of the second ontology.

3. Similarity Computation The similarity computation between an entity of O_1 and an entity of O_2 is done by using a wide range of similarity functions. Each similarity function is based on a feature (Section 4.1) and the respective similarity measure (Section 4.2). For NOM they are shown in Table 2.

4. Similarity Aggregation NOM does not aggregate the individual similarity results by a linear combination, but before aggregation it employs a function that emphasizes high individual similarities and de-emphasizes low individual similarities e.g. a sigmoid function. To produce an aggregated similarity (cf. Section 4.2) NOM applies $adj(x) = \frac{1}{1+e^{-5(x-0.5)}}$. Weights w_k are assigned by maximizing the f-measure on training data.

5. Interpretation NOM interpretes similarity results by two means. First, it applies a threshold to discard spurious evidence of similarity. Second, NOM enforces bijectivity

	No.	Feature	Measure
Concept Similarity	1	label	string similarity
	2	URI	string equality
	3	sameAs relation	explicit equality
	4	direct properties	SimSet
	5	all inherited properties	SimSet
	6	all super-concepts	SimSet
	7	all sub-concepts	SimSet
	8	concept siblings	SimSet
	9	direct instances	SimSet
	10	instances	SimSet
Relation Similarity	1	label	string similarity
	2	URI	string equality
	3	sameAs relation	explicit equality
	4	domain and range	object equality
	5	all super-properties	SimSet
	6	all sub-properties	SimSet
	7	property siblings	SimSet
	8	property instances	SimSet
Instance Similarity	1	label	string similarity
	2	URI	string equality
	3	sameAs relation	explicit equality
	4	all parent-concepts	SimSet
	5	property instances	SimSet
Property-instance Similarity	1	domain and range	object equality
	2	parent property	SimSet

Table 2. NOM: Features and Measures for Similarity

of the mapping by ignoring candidate pairs that would violate this constraint and by favoring candidate pairs with highest aggregated similarity scores.

6. Iteration The first round uses only the basic comparison method based on labels (Function no. 1) to compute the similarity between entities. By doing the computation in several rounds one can access the already computed pairs and use more advanced overall similarities. Therefore, in the second round and thereafter NOM relies on all the similarity functions shown in table 2.

5.2 PROMPT

PROMPT is a semi-automatic tool described in [2]. It was one of the first tools for ontology merging. For this paper we concentrate on the actions performed to identify possible mapping candidates aka. merging candidates. For this PROMPT does not require all of the steps of the process model.

1. Feature Engineering As a plug-in to Protege, PROMPT uses the triples of RDFS as features.

2. Search Step Selection Like NOM, PROMPT relies on a complete comparison. Each pair of entities from ontology one and two is checked for their similarities.

3. Similarity Computation The system determines the similarities based on whether entities have similar labels. Specifically, PROMPT checks for identical labels. This is a further restriction compared to our similarity function no. 1, which also allows small deviations in the spelling.

5. Interpretation PROMPT presents the pairs with a similarity above a defined threshold threshold and therefore high probability of actually being entities for merging to the user. From these propositions, the user chooses the correct ones which are, in the case of PROMPT, merged.

6. Iteration Iteration is done in PROMPT to allow manual refinement. After the user has acknowledged the proposition, the system recalculates the corresponding similarities and comes up with new merging suggestions.

5.3 Anchor-PROMPT

Anchor-PROMPT represents an advanced version of PROMPT which includes similarity measures based on ontology structures.

3. Similarity Computation Anchor-PROMPT traverses paths between anchor points (entity pairs already identified as equal). Along these paths new mapping candidates are suggested. Specifically, paths are traversed along hierarchies and along relations.

5.4 GLUE

GLUE [3] uses machine learning techniques to achieve good mappings.

1. Feature Engineering In a first step the Distribution Estimator uses a multi-strategy machine learning approach based on a sample mapping set. It learns a strategy to identify equal instances and concepts.

2. Search Step Selection Also GLUE checks every entity pair.

3. Similarity Computation, 4. Similarity Aggregation The Similarity Estimator determines the similarity of two instances based on the learnt rules. From this also the mapping of concepts is derived.
Concepts and relations are further compared using Relaxation Labelling. The intuition of Relaxation Labelling is that the label of a node (in our terminology: mapping assigned to an entity) is typically influenced by the features of the node's neighborhood in the graph. The authors explicitly mention subsumption, frequency, and "nearby" nodes.A local optimal mapping for each entity is determined using the similarity results of neighboring entity pairs from a previous round.
Normally one would have to check all possible labelling configurations, which includes the mappings of all other entities. The developers are well aware of the problem arising in complexity, so they set up sensible partitions with the same features in the labelling configurations.

5. Interpretation From the previous step we receive the probabilities of two entities mapping onto each other. The maximum probable pair is the final mapping result.

6. Iteration Simply to gain meaningful results the relaxation labelling step has to be repeated several times.

5.5 QOM - Quick Ontology Mapping

The goal of this paper is to present an efficient mapping algorithm. We propose the Quick Ontology Mapping (QOM) approach for this purpose. Our proceeding it to check each process step of the effective but inefficient NOM approach for measures that could raise efficiency.

1. Feature Engineering Like NOM, the ontologies for the QOM approach have to be given in RDFS triples.

2. Search Step Selection A major problem in run-time complexity is due to the number of candidate mapping pairs which have to be compared to actually find the best mappings. We try to minimize this number.

In particular we use a dynamic programming approach [11]. In this approach we have two main data structures, i.e. problem descriptions. First we have the analysis of entity pairs which ought to be compared. And second an agenda putting the individual analyses into a meaningful (and efficient) order in which they are executed. Each analysis then may or may not initiate a new problem description which is then aligned in the agenda. The newly created problem descriptions depend on the analysis strategy that is pursued. The approach we pursue corresponds to a beam search in which we try to focus on the most promising comparisons. This is done at the cost of losing completeness of the analysis.

To focus on the most promising entity pairs we consider ontology structures that direct the beam effectively and efficiently. QOM pursues the subsequent strategies strategies to create new entity pair analyses: A naive approach is to choose a fixed number (or percentage) of *random entity pairs* from all possible pairs. As the duration of one loop in the process is shorter, we receive results earlier than it would be the case for complete comparisons. Second *labels* play a major role in the similarity process. The problem description therefore only contains entities whose labels are close to each other. Practically the entity labels are placed into a sorted list. For every entity in the list we schedule the neighbors which have the same first three letter for the comparison analysis. Third we compare only entities for which adjacent entities were assigned new mappings in a previous iteration (*Mapping Change Propagation*). Our *combined* approach makes use of different optimization strategies: we use the label agenda, the random agenda, and the mapping change propagation agenda. During the iterations the focus shifts from labels, which are good to identify the first mappings, to random, which is the only way to find mappings if no other relations lead to the mappings. A number of other strategies (e.g. based on the taxonomy, already identified mappings) or combinations of strategies were pretested with very simple data sets but didn't return promising results.

3. Similarity Computation In order to implement an efficient and effective strategy we trade-off between the efficiency of single methods and their coverage. After having identified the presumably most costly methods, we try to lower their cost or eliminate them. In particular, we remove rules which compare whole sub-trees and replace them

through rules which access only a limited number of entities for the comparison, as one can see in Table 3. All other rules are maintained from the NOM approach.

	Change	Feature	Measure
Concept Similarity	$\longrightarrow$ 5a	properties of super-concepts	SimSet
	$\longrightarrow$ 6a	direct super-concepts	SimSet
	$\longrightarrow$ 7a	direct sub-concepts	SimSet
	$\longrightarrow$ 10a	instances of sub-concepts	SimSet
Relation Similarity	$\longrightarrow$ 5a	direct super-properties	SimSet
	$\longrightarrow$ 6a	direct sub-properties	SimSet
Instance Similarity	$\longrightarrow$ 4a	direct parent-concepts	SimSet

Table 3. Changed Rules and Complexity

4. Similarity Aggregation The aggregation of single methods is the same as in the NOM approach.

5. Interpretation Also the interpretation steps remains. A threshold is determined and bijectivity has to be ensured.

6. Iteration We also retain the multiple rounds. In all our tests we find that after five to ten rounds hardly any further changes occur to the mapping table. We therefore restrict the number of runs to ten.

6 Comparing Run-time Complexity

We determine the worst case run-time complexity of the mapping proposing algorithms depending on the size of the two given ontologies. Thereby, we wanted to base our analysis on realistic ontologies and not on artifacts, e.g. we wanted to avoid the consideration of large ontologies with n leaf concepts but a depth of the concept hierarchy H_C of $n-1$. For this purpose, we constrain our ontologies to have parameter settings like the ones found in [12]. They have examined the structure of a large number of ontologies and found, e.g., that concept hierarchies typically have a branching factor b of around 2. They also found that the concept hierarchies are neither extremely shallow nor extremely deep. Hence, we base all subsequent results on such constraints.

Theorem 1. *The worst case run-time behaviors of NOM, PROMPT, Anchor-PROMPT, GLUE and QOM are given by the following table:*

NOM	$O(n^2 \cdot log^2(n))$	
PROMPT	$O(n^2)$	
Anchor-PROMPT	$O(n^2 \cdot log^2(n))$	
GLUE	$O(n^2)$	*Note: This result is based on optimistic assumptions about the learner.*
QOM	$O(n \cdot log(n))$	

Proof Sketch 1 *The following subjects represent a proof sketch for the complexities of the specific approaches.*

The number of entities involved in the feature and the complexity of the respective similarity measure eventually affect the run-time performance of the algorithm.

*The single elements of complexity are aligned to the canonical process of Section 3. First we determine the cost for feature engineering (*feat*). The second step is the selection of search steps (*sel*). The actual number of search steps i.e. candidate mapping pair comparisons is represented through* comp*. For each of these comparisons we need to compute* k *different similarity functions* sim_k *and aggregate them (*agg*). From this the interpretation of the similarity results with respect to mapping requires a run-time complexity of* inter*. Finally we have to iterate over the previous steps multiple times (*iter*).*

The worst case run-time complexity is defined for all approaches by:

$$c = (feat + sel + comp \cdot (\sum_k sim_k + agg) + inter) \cdot iter$$

Depending on the concrete values that show up in the individual process steps the different run-time complexities are derived in detail in [6].

In our considerations we do not regard the complexity due to storage and access of entities in a repository.

7 Empirical Evaluation and Results

In this section we show that the worst case considerations carry over to practical experiments and that the quality of QOM is only negligibly lower than the one of other approaches. The implementation itself was coded in Java using the KAON-framework[4] for ontology operations.

7.1 Test Scenario

Metrics We ensure comparability not only between our own test series, but also with existing literature. Therefore we focus on using standard information retrieval metrics to measure our results. The metrics we measure over time are defined as follows (cf. [13]):

Precision $p = \frac{\#correct\_found\_mapping}{\#found\_mappings}$
Recall $r = \frac{\#correct\_found\_mappings}{\#existing\_mappings}$
F-Measure $f_1 = \frac{2pr}{p+r}$

Data Sets Three separate data sets were used for evaluation purposes. As real world ontologies and especially their mappings are scarce, students were asked to independently create and map ontologies.

Russia 1 In this first set we have two ontologies describing Russia. The students created the ontologies with the task to represent the content of two independent travel websites

[4] http://kaon.semanticweb.org/

about Russia. These ontologies have approximately 400 entities each, including concepts, relations, and instances. The total number of possible mappings is 160, which the students have assigned manually.

Russia 2 The second set again covers Russia. This time the two further ontologies are more difficult to map. The differ substantially in both labels and structure. Each ontology has about 300 entities with 215 possible mappings, which were already captured during generation.

Tourism Finally, the participants of a seminar created two ontologies which separately describe the tourism domain of Mecklenburg-Vorpommern. Both ontologies have an extent of about 500 entities. No instances were modelled with this ontology though, they only consist of concepts and relations. The 300 mappings were created manually.

Strategies We evaluated three mapping strategies:

- NOM - Naive Ontology Mapping: NOM is an approach making use of a wide range of features and measures. Therefore it reaches high levels of effectivity and represents our quality baseline. In terms of complexity it is similar to the described Anchor-PROMPT approach.
- LOM - Labels-only Ontology Mapping: LOM basically corresponds to the PROMPT approach. As this approach is rather simple and fast we use it as a baseline to evaluate the speed. Please note that the string matching measure in LOM uses our string similarity based on Levenshtein, whereas PROMPT uses exact string comparisons.
- QOM - Quick Ontology Mapping: QOM is our new approach focusing on efficiency. It uses an agenda of combined strategies as well as several other optimizing measures as described.

These strategies were chosen because only for them we could ensure similar starting positions. Specifically, our three approaches are fully automatic (whereas PROMPT and Anchor-PROMPT are semi-automatic), and do not need large training sets to begin with (as GLUE needs). We will shortly make both software and data sets available on our website for the interested readers. Researchers are welcome to enhance and re-use them.

7.2 Results and Discussion

We present the results of the strategies on each of the three data sets in Figure 3. The x-axis shows the elapsed time on a logarithmic scale, the y-axis corresponds to the reached f-measure on correct mappings. The dots represent the result after each iteration step.

The f-measure value of the NOM strategy rises very slowly but reaches high absolute values. Unfortunately it requires a lot of calculation time. LOM reaches good results in much shorter time. Please notice that for ontologies with a small number of similar labels (Figure 3, graph 2) this strategy is not satisfactory. Finally the QOM Strategy is plotted. It reaches high quality levels very fast. In terms of absolute values it also seems to reach the best quality results of all strategies. This is an effect of having the largest amount of rounds.

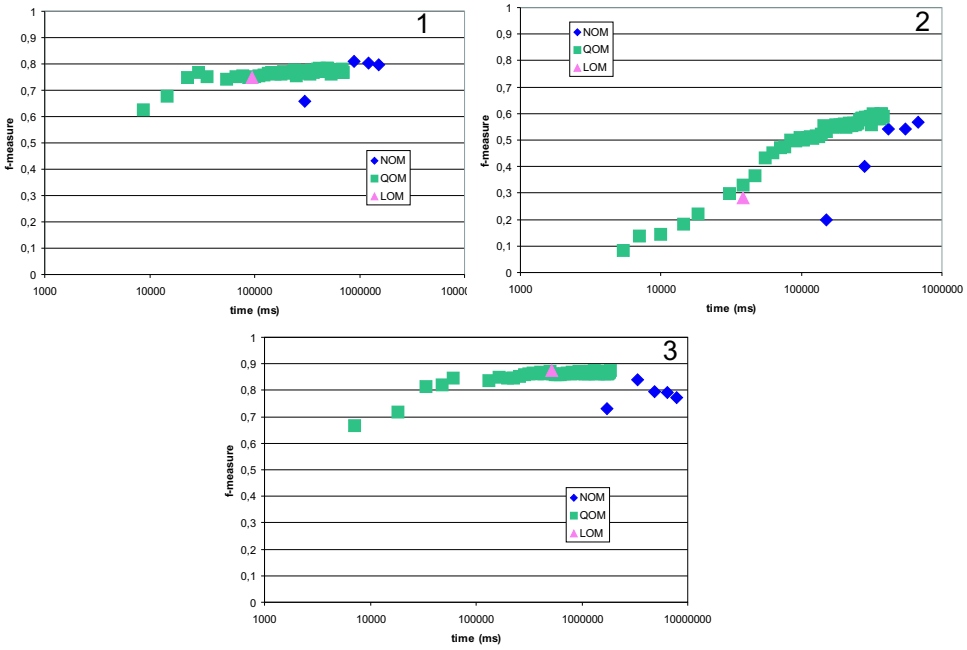

Fig. 3. Mapping quality reached over time with Russia 1 [1], Russia 2 [2], and Tourism [3] ontologies.

This paper focuses on comparing efficient algorithms with naive ones. The absolute f-measure values are often lower than one might expect, but one has to keep in mind that some of the mappings may not even be determined by humans.

We had the hypothesis that faster mapping results can be obtained with only a negligible loss of quality.

- The first fact is that optimizing for efficiency lowers the overall quality. If ontologies are not too large we should refrain from doing this.
- Labels are very important for mapping, if not the most important feature of all, and alone return very satisfying results.
- The Quick Ontology Mapping approach shows equally good results than the slower strategies, effectively not lowering quality.
- QOM is with a factor of 10 to 100 times faster, depending on the scenario.

Recapitulating we can say that making use of ontology specific features allows to build mapping approach which is both very effective and efficient.

8 Related Work

Various authors have tried to find a general description of similarity with several of them being based on knowledge networks. [14] give a general overview of similarity. As the

basic ontology mapping problem has been around for some years first tools have already been developed to address this. The tools PROMPT and AnchorPROMPT [2] use labels and to a certain extent the structure of ontologies. However, their focus lies on ontology merging i.e. how to create one ontology out of two. Potential matches are presented to the user for confirmation. In their tool ONION [15] the authors take up the idea of using rules and inferencing for mapping, but the inferencing is based on manually assigned mappings or simple heuristics (as e.g. label comparisons). [3] already use a general approach of relaxation labelling in their tool GLUE. However, most of their work is based on the similarity of instances only. Besides equality first steps are taken in the direction of complex matches. These could also include concatenation of two fields such as “first name” and “last name” to “name”[16]. [17] further present an approach for semantic mappings based on SAT. Recapitulating, despite the large number of related work, there are very few approaches raising the issue of efficiency.

To get a broader view, especially for ideas on efficiency considerations, we name related areas. [18] express their insights from a database view. Many ideas from the database community, especially concerning efficiency (see [19]), also have to be regarded for ontologies. However, even though these algorithms have been optimized for many years, they can only be partly used for our purposes, as they are mainly oriented towards (domain-specific) instance comparisons rather than schema matching. Another community involved in similarity and mapping are object-oriented representations[20] in which surprisingly little work has been done. Finally, agent communication greatly benefits from mapping, and has therefore been involved in the topic (e.g. [21]). Even though efficiency has been a topic in related areas, only very little can directly be transferred to ontology mapping.

9 Conclusion

The mapping problem arises in many scenarios. We have shown a methodology for identifying mappings between two ontologies based on the intelligent combination of ontology features and similarity measures. Our focus was to show that the whole ontology mapping process can be done efficiently. We presented our mapping process QOM based on ontology features and similarity measures. Additionally we made use of ontology specific features to create an efficient approach. Theoretical considerations and practical evaluation proved our initial hypothesis. QOM is equal effective than standard time consuming approaches, but was achieved much more efficiently. It was by an order of magnitude faster in theory and 10 to 100 times faster in empirical evaluations. We expect that the good results have an impact on applications relying on ontology mapping.

Acknowledgements Research reported in this paper has been partially financed by the EU in the IST projects SWAP (IST-2001-34103) and SEKT (IST-2003-506826).

References

1. Agrawal, R., Srikant, R.: On integrating catalogs. In: Proceedings of the tenth international conference on World Wide Web, ACM Press (2001) 603–612

2. Noy, N.F., Musen, M.A.: The PROMPT suite: interactive tools for ontology merging and mapping. International Journal of Human-Computer Studies **59** (2003) 983–1024
3. Doan, A., Domingos, P., Halevy, A.: Learning to match the schemas of data sources: A multistrategy approach. VLDB Journal **50** (2003) 279–301
4. Ehrig, M., Haase, P., van Harmelen, F., Siebes, R., Staab, S., Stuckenschmidt, H., Studer, R., Tempich, C.: The SWAP data and metadata model for semantics-based peer-to-peer systems. In: Proceedings of MATES-2003. First German Conference on Multiagent Technologies. LNAI, Erfurt, Germany, Springer (2003)
5. Hotho, A., Staab, S., Stumme, G.: Ontologies improve text document clustering. In: Proceedings of the International Conference on Data Mining — ICDM-2003, IEEE Press (2003)
6. Ehrig, M., Staab, S.: Quick ontology mapping with QOM. Technical report, University of Karlsruhe, Institute AIFB (2004) http://www.aifb.uni-karlsruhe.de/WBS/meh/mapping/.
7. Bisson, G.: Why and how to define a similarity measure for object based representation systems. Towards Very Large Knowledge Bases (1995) 236–246
8. Maedche, A., Staab, S.: Measuring similarity between ontologies. In: Proceedings of the European Conference on Knowledge Acquisition and Management (EKAW), Springer (2002)
9. Levenshtein, I.V.: Binary codes capable of correcting deletions, insertions, and reversals. Cybernetics and Control Theory (1966)
10. Cox, T., Cox, M.: Multidimensional Scaling. Chapman and Hall (1994)
11. Boddy, M.: Anytime problem solving using dynamic programming. In: Proceedings of the Ninth National Conference on Artificial Intelligence, Anaheim, California, Shaker Verlag (1991) 738–743
12. Tempich, C., Volz, R.: Towards a benchmark for semantic web reasoners - an analysis of the DAML ontology library. In Sure, Y., ed.: Evaluation of Ontology-based Tools (EON2003) at Second International Semantic Web Conference (ISWC 2003). (2003)
13. Do, H., Melnik, S., Rahm, E.: Comparison of schema matching evaluations. In: Proceedings of the second int. workshop on Web Databases (German Informatics Society). (2002)
14. Rodrguez, M.A., Egenhofer, M.J.: Determining semantic similarity among entity classes from different ontologies. IEEE Transactions on Knowledge and Data Engineering (2000)
15. Mitra, P., Wiederhold, G., Kersten, M.: A graph-oriented model for articulation of ontology interdependencies. Lecture Notes in Computer Science **1777** (2000) 86+
16. Do, H., Rahm, E.: COMA - a system for flexible combination of schema matching approaches. In: Proceedings of the 28th VLDB Conference, Hong Kong, China (2002)
17. Bouquet, P., Magnini, B., Serafini, L., Zanobini, S.: A SAT-based algorithm for context matching. In: IV International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT'2003), Stanford University (CA, USA) (2003)
18. Roddick, J., Hornsby, K., de Vries, D.: A unifying semantic distance model for determining the similarity of attribute values. In: Proceedings of the 26th Australsian Computer Science Conference (ACSC2003), Adelaide, Australia (2003)
19. McCallum, A., Nigam, K., Ungar, L.H.: Efficient clustering of high-dimensional data sets with application to reference matching. In: Knowledge Discovery and Data Mining. (2000) 169–178
20. Bergmann, R., Stahl, A.: Similarity measures for object-oriented case representations. Lecture Notes in Computer Science **1488** (1998) 25+
21. Weinstein, P., Birmingham, W.P.: Comparing concepts in differentiated ontologies. In: Proceedings of the Twelfth Workshop on Knowledge Acquistion, Modeling and Management (KAW'99), Banff, Alberta, Canada (1999)