Efficiency of Ontology Mapping Approahces

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 experiments 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 against 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 marginal, but the improvement of efficiency is so tremendous that it allows for the

[1] Frequently also called alignment.

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 analyze 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 ontologies. 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:*

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

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

with $\forall e \in O_1(\exists f \in O_2 : \mathrm{map}(e, O_1, O_2) = f \vee \mathrm{map}(e, O_1, O_2) = \perp)$.

An entity can either be 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 $\mathrm{map}_{O_1,O_2}(e)$ for $\mathrm{map}(e, O_1, O_2)$. We derive a relation map_{O_1,O_2} by defining $\mathrm{map}_{O_1,O_2}(e, f) \Leftrightarrow \mathrm{map}_{O_1,O_2}(e) = f$. We leave out O_1, O_2 when they are evident from the context and write $\mathrm{map}(e) = f$ and $\mathrm{map}(e, f)$, 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 $\mathrm{map}(e, f)$. 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.

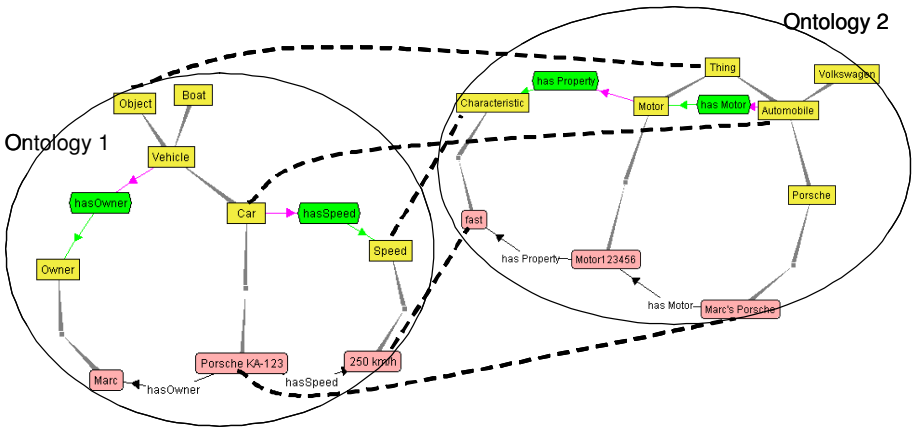

Fig. 1. Example Ontologies and their Mappings

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 $\mathrm{map}_{O_1,O_2}(e, f)$

Apart from one-to-one mappings as investigated in this paper one entity often has to be mapped to a complex composite such as a concatenation of terms (first and last name) or an entity with restrictions (a sports-car is a car going faster than 250 km/h). We refer to [7] for adequate methods.

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}.

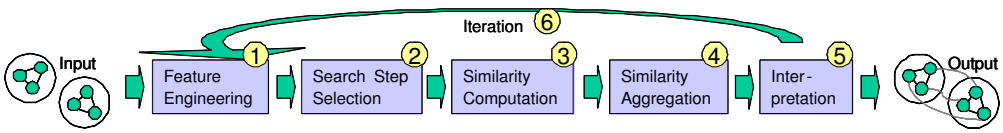

Fig. 2. Mapping Process

[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.

4.1 Features of Ontological Entities

To compare two entities from two different ontologies, one considers some of their characteristics, i.e. their features. The features may be specific for a mapping generation algorithm, in any case the features of ontological entities (of concepts, relations, instances) need to be extracted from extensional and intensional ontology definitions. Then an entity is described by the kind of appearance that is found to hold for this entity for characteristics like:

- *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 (e.g., 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 usage of the sameAs relation
- *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

Example We again refer to the example in Figure 1. The actual feature consists of a juxtaposition of relation name and entity name. The Car concept of ontology 1 is characterized e.g. through its (label, Car), the concept which it is linked to through (super concept, Vehicle) resp. (subclassOf, Vehicle), its (concept sibling, boat), (direct property, hasSpeed), and the instances linked through ((instance, Porsche KA-123) resp. (type-of, Porsche KA-123). The relation hasSpeed on the other hand is described through e.g. the (domain, Car) and the (range, Speed). An instance would be Porsche KA-123, which is characterized e.g. through the instantiated (property instance, (hasOwner, Marc)), and the instantiated (property instance, (hasSpeed, 250 km/h)).

4.2 Similarity Computation

Definition 2. *We define a similarity measure for comparison of ontology entities as a function as follows (cf. [8]):*

$$\mathrm{sim} : \mathcal{E} \times \mathcal{E} \times \mathcal{O} \times \mathcal{O} \rightarrow [0, 1]$$

Different similarity measures $\mathrm{sim}_k(e, f, O_1, O_2)$ are indexed through a label k. Further, we leave out O_1, O_2 when they are evident from the context and write $\mathrm{sim}_k(e, f)$. The following similarity measures are needed to compare the features of ontological entities.

- *Object Equality* is based on existing logical assertions — especially assertions from previous iterations: $\mathrm{sim}_{obj}(a, b) := \{1 \text{ iff } map_{prev}(a) = b, 0 \text{ otherwise}\}$
- *Explicit Equality* checks whether a logical assertion already forces two entities to be equal: $\mathrm{sim}_{exp}(a, b) := \{1 \text{ iff } statement(a, \text{“}sameAs'', b), 0 \text{ otherwise}\}$
- *String Similarity* measures the similarity of two strings on a scale from 0 to 1 (cf. [9]) based on Levenshtein's edit distance, ed [10].
  $\mathrm{sim}_{str}(c, d) := max(0, \frac{min(|c|,|d|) - ed(c,d)}{min(|c|,|d|)})$
- *SimSet*: For many features we have to determine to what extent two sets of entities are similar. To remedy the problem multidimensional scaling [11] 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:
  $\mathrm{sim}_{set}(E, F) = \frac{\sum_{e \in E} \boldsymbol{e}}{|E|} \cdot \frac{\sum_{f \in F} \boldsymbol{f}}{|F|}$
  with $\boldsymbol{e} = (\mathrm{sim}(e, e_1), \mathrm{sim}(e, e_2), \ldots, \mathrm{sim}(e, f_1), \mathrm{sim}(e, f_2), \ldots)$, $\boldsymbol{f}$ analogously.

These measures are all input to the similarity aggregation.

4.3 Similarity Aggregation

Similarities are aggregated by:
Aggregated Similarity: $\mathrm{sim}_{agg}(e, f) = \frac{\sum_{k=1\ldots n} w_k \cdot \mathrm{adj}(\mathrm{sim}_k(e,f))}{\sum_{k=1\ldots n} w_k}$
with w_k being the weight for each individual similarity measure, and adj being a function to transform the original similarity value ($adj : [0, 1] \rightarrow [0, 1]$) such as the sigmoid function.

4.4 Interpretation

From the similarity values we derive the actual mappings. The basic idea is that each entity may only participate in one mapping and that we assign mappings based on a greedy strategy that starts with the largest similarity values first. Ties are broken arbitrarily.

$$P(g, h, U \backslash \{e\}, V \backslash \{f\}) \leftarrow P(e, f, U, V) \wedge \mathrm{sim}(g, h) > t$$
$$\wedge (g, h) \in U \backslash \{e\} \times V \backslash \{f\} = \mathrm{argmax}_{(g,h)} sim(g, h).$$
$$P(\bot, \bot, E \cup \{\bot\}, E \cup \{\bot\}).$$
$$\mathrm{map}(e, f) \leftarrow \exists X_1, X_2 P(e, f, X_1, X_2) \wedge \neg(e, f) = (\bot, \bot).$$

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, the ontologies have to be represented RDFS. The features are required in a way shown in Section 4.1.

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) of both ontologies and the respective similarity measure (Section 4.2). The corresponding ontology is indicated through an index. For NOM they are shown in Table 2.

Comparing	No.	Feature	Similarity Measure
Concepts	1	(label,X_1)	string similarity(X_1, X_2)
	2	(URI_1)	string equality(URI_1, URI_2)
	3	(X_1,sameAs,X_2) relation	explicit equality(X_1, X_2)
	4	(direct properties,Y_1)	SimSet(Y_1, Y_2)
	5	all (inherited properties,Y_1)	SimSet(Y_1, Y_2)
	6	all (super-concepts,Y_1)	SimSet(Y_1, Y_2)
	7	all (sub-concepts,Y_1)	SimSet(Y_1, Y_2)
	8	(concept siblings,Y_1)	SimSet(Y_1, Y_2)
	9	(direct instances,Y_1)	SimSet(Y_1, Y_2)
	10	(instances,Y_1)	SimSet(Y_1, Y_2)
Relations	1	(label,X_1)	string similarity(X_1, X_2)
	2	(URI_1)	string equality(URI_1, URI_2)
	3	(X_1,sameAs,X_2) relation	explicit equality(X_1, X_2)
	4	(domain,X_d1) and (range,X_r1)	object equality(X_{d1}, X_{d2}) and (X_{r1}, X_{r2})
	5	all (super-properties,Y_1)	SimSet(Y_1, Y_2)
	6	all (sub-properties,Y_1)	SimSet(Y_1, Y_2)
	7	(property siblings,Y_1)	SimSet(Y_1, Y_2)
	8	(property instances,Y_1)	SimSet(Y_1, Y_2)
Instances	1	(label,X_1)	string similarity(X_1, X_2)
	2	(URI_1)	string equality(URI_1, URI_2)
	3	(X_1,sameAs,X_2) relation	explicit equality(X_1, X_2)
	4	all (parent-concepts,Y_1)	SimSet(Y_1, Y_2)
	5	(property instances,Y_1)	SimSet(Y_1, Y_2)
Property-instances	1	(domain,X_d1) and (range,X_r1)	object equality(X_{d1}, X_{d2}) and (X_{r1}, X_{r2})
	2	(parent property,Y_1)	SimSet(Y_1, Y_2)

Table 2. Features and Similarity Measures for Different Entity Types Contributing to Aggregated Similarity in NOM

4. Similarity Aggregation NOM emphasizes high individual similarities and de-emphasizes low individual similarities by weighting individual similarity results with

a sigmoid function first and summing the modified values then. 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 of the mapping by ignoring candidate mappings that would violate this constraint and by favoring candidate mappings with highest aggregated similarity scores.

6. Iteration The first round uses only the basic comparison method based on labels and string similarity to compute the similarity between entities. By doing the computation in several rounds one can access the already computed pairs and use more sophisticated structural similarity measures. Therefore, in the second round and thereafter NOM may rely on all the similarity functions listed 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 RDFS with features as in the previous approach.

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 string similarity, which also allows small deviations in the spelling.

4. Similarity Aggregation As PROMPT uses only one similarity measure, aggregation is not necessary.

5. Interpretation PROMPT presents the pairs with a similarity above a defined threshold. For these pairs chances are high that they are merged by the user. The user selects the ones he deems to be correct, which are then merged in PROMPT.

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. Only the similarity computation (step 3) changes.

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 as well as along other relations.

5.4 GLUE

GLUE [3] uses machine learning techniques to determine 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 GLUE checks every candidate mapping.

3. Similarity Computation, 4. Similarity Aggregation, 5. Interpretation 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 i.e. labelling sets with the same features are grouped and processed only once.
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 To gain meaningful results only the relaxation labelling step and its interpretation have to be repeated several times. The other steps are just carried out once.

5.5 QOM — Quick Ontology Mapping

The goal of this paper is to present an efficient mapping algorithm. For this purpose, we optimize the effective, but inefficient NOM approach towards efficiency. The outcome is QOM — Quick Ontology Mapping.

1. Feature Engineering Like NOM, QOM exploits RDF 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. Therefore, we use heuristics to lower the number of candidate mappings.

In particular we use a dynamic programming approach [12]. In this approach we have two main data structures. First, we have candidate mappings which ought to be investigated. Second, an agenda orders the candidate mappings in a meaningful and efficient way, discarding some of them entirely. The completed analysis of a candidate mapping may or may not bring a new candidate mapping onto the agenda. The behavior of initiative and ordering constitutes a search strategy. The strategy QOM pursues corresponds to a beam search that focuses on the most promising candidate mappings — possibly missing some good candidate mappings altogether.

To focus on the most promising candidate mappings we consider ontology structures that direct the beam effectively and efficiently. QOM combines the subsequent

strategies to propose new candidate mappings for inspection: First, QOM *randomly* selects a fixed number (or percentage) of candidate mappings from all possible mappings. Second, QOM restricts candidate mappings to entity pairs whose *labels* are near to each other in a sorted list, i.e. that have the same first three letters. Third, QOM compares only entities for which adjacent entities were assigned new mappings in a previous iteration (*Mapping Change Propagation*). The *combined* approach of QOM makes use of different optimization strategies: it uses a label subagenda, a randomness subagenda, and a mapping change propagation subagenda. In the first iteration the label subagenda is pursued. Afterwards we focus on mapping change propagation. Finally we shift to the randomness subagenda, if the other strategies do not sufficiently identify correct mapping candidates.[4] With this agenda we only have to check a fixed and restricted number of mapping candidates for each original entity.

3. Similarity Computation In order to optimize NOM towards QOM, we have restricted the range of costly features as specified in Table 3. In particular, QOM avoids the complete pair-wise comparison of trees in favor of a(n incomplete) top-down strategy. All other features are maintained from NOM.

Comparing	Change	Feature	Similarity Measure
Concepts	$\longrightarrow$ 5a	(properties of direct super-concepts, Y_1)	SimSet(Y_1, Y_2)
	$\longrightarrow$ 6a	(direct super-concepts, Y_1)	SimSet(Y_1, Y_2)
	$\longrightarrow$ 7a	(direct sub-concepts, Y_1)	SimSet(Y_1, Y_2)
	$\longrightarrow$ 10a	(instances of direct sub-concepts, Y_1)	SimSet(Y_1, Y_2)
Relations	$\longrightarrow$ 5a	(direct super-properties, Y_1)	SimSet(Y_1, Y_2)
	$\longrightarrow$ 6a	(direct sub-properties, Y_1)	SimSet(Y_1, Y_2)
Instances	$\longrightarrow$ 4a	(direct parent-concepts, Y_1)	SimSet(Y_1, Y_2)

Table 3. Features and Similarity Measurs used in QOM

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

5. Interpretation Also the interpretation step of QOM is the same as in NOM. A threshold is determined and bijectivity of mappings is maintained.

6. Iteration QOM iterates to find mappings based on lexical knowledge first and based on knowledge structures later. In all our tests we have found that after five to ten rounds hardly any further changes occur in the mapping table. QOM therefore restricts the number of runs to ten.

[4] We have also explored a number of other strategies (e.g. based on the taxonomy, already identified mappings) or combinations of strategies with simple data sets but they did not outperform results of QOM presented here.

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 [13]. They have examined the structure of a large number of ontologies and found, e.g., that concept hierarchies typically have a branching factor of around 2 and that the concept hierarchies are neither extremely shallow nor extremely deep. Hence, in the following we base our results on their findings.

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[5]	$O(n^2)$
QOM	$O(n \cdot log(n))$

Proof Sketch 1 *The different algorithmic steps contributing to complexity are aligned to the canonical process of Section 3.*

*For each of the algorithms, one may then determine the costs of each step. First, one determines the cost for feature engineering (*feat*). The second step is the process of search step i.e. candidate mappings selection (*sele*). For each of the selected candidate mappings (*comp*) we need to compute* k *different similarity functions* sim_k *and aggregate them (*agg*). The number of entities involved and the complexity of the respective similarity measure affect the run-time performance. Subsequently the interpretation of the similarity values with respect to mapping requires a run-time complexity of* inter. *Finally we have to iterate over the previous steps multiple times (*iter*).*

Then, the worst case run-time complexity is defined for all approaches by:

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

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

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[6] for ontology operations.

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

7.1 Test Scenario

Metrics We use standard information retrieval metrics to assess the different approaches (cf. [14]):

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 objectives to represent the content of two independent travel websites 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 ontologies are more difficult to map. They differ substantially in both labels and structure. Each ontology has about 300 entities with 215 possible mappings, which were 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 the mapping strategies described in the previous sections:

- PROMPT — As the PROMPT algorithm is rather simple and fast we use it as a baseline to evaluate the speed.
- NOM / Anchor-PROMPT — Naive Ontology Mapping is an approach making use of a wide range of features and measures. Therefore it reaches high levels of effectiveness and represents our quality baseline. In terms of structural information used and complexity incurred it is somewhat similar to Anchor-PROMPT.
- QOM — Quick Ontology Mapping: QOM is our novel approach focusing on efficiency.

To circumvent the problem of having semi-automatic merging tools (PROMPT and Anchor-PROMPT) in our fully automatic mapping tests, we assumed that every proposition of the system is meaningful and correct. Further, as we have difficulties in running Anchor-PROMPT with the size of the given data sets, we refer to the results of the somewhat similar NOM. For GLUE we face another general problem. The algorithm has a strong focus on example instance mappings. As we can not provide this, we refrained from running the tests on a lowly trained estimator which would immediately result in poor quality results.

For the interested readers we will shortly make available both software and data sets on our website. Researchers will be welcome to enhance and re-use them.

7.2 Results and Discussion

We present the results of the strategies on each of the data sets in Figures 3 to 4. The tourism dataset shows similar characteristics as Russia 1 and is therefore not plotted. The x-axis shows the elapsed time on a logarithmic scale, the y-axis corresponds to the f-measure. The symbols represent the result after each iteration step.

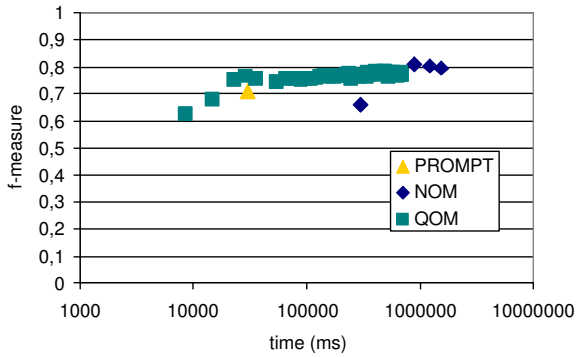

Fig. 3. Mapping quality reached over time with Russia 1 ontologies.

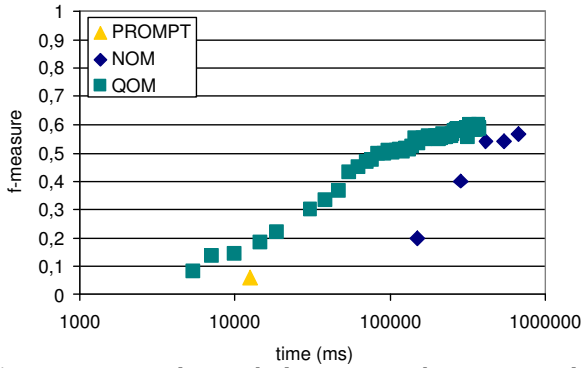

Fig. 4. Mapping quality reached over time with Russia 2 ontologies.

Depending on the scenario PROMPT reaches good results within a short period of time. Please notice that for ontologies with a small number of similar labels (Figure 4) this strategy is not satisfactory (f-measure 0.06). In contrast, the f-measure value of the NOM strategy rises slowly but reaches high absolute values of up to 0.8. Unfortunately it requires a lot of time. Finally the QOM Strategy is plotted. It reaches high quality levels very quickly. In terms of absolute values it also seems to reach the best quality results of all strategies. This appears to be an effect of QOM achieving an about 20 times higher number of iterations than NOM within the given time time frame.

Lessons Learned. We had the hypothesis that faster mapping results can be obtained with only a negligible loss of quality. We here briefly present the bottom line of our considerations in this paper:

1. Optimizing the mapping approach for efficiency — like QOM does — decreases the overall mapping quality. If ontologies are not too large one might prefer to rather avoid this.
2. Labels are very important for mapping, if not the most important feature of all, and alone return very satisfying results.
3. Using an approach combining many features to determine mappings clearly leads to significantly higher quality mappings.
4. The Quick Ontology Mapping approach shows very good results. Quality is lowered only marginally.
5. QOM is with a factor of 10 to 100 times faster than standard prominent approaches.

Recapitulating we can say that our mapping approach is 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. [15] 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 [16] the authors use rules and inferencing to execute mappings, but the inferencing is based on manually assigned mappings or heuristics simpler than PROMPT. [3] 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”[7]. [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.

Apart from the ontology domain research on mapping and integration has been done in various computer science fields. [1] present an approach to integrate documents from different sources into a master catalog. As schema and instance integration have been a topic within the database community we like to refer to their related approaches in [18] and [19]), which already deal with efficiency.

9 Conclusion

The problem of mapping two ontologies effectively and efficiently arises in many application scenarios [4, 5]. We have devised a generic process model to investigate and compare different approaches that generate ontology mappings. In particular, we have developed an original method, QOM, for identifying mappings between two ontologies. We have shown that QOM is on a par with other good state-of-the-art algorithms concerning the quality of proposed mappings, while outperforming them with respect to efficiency — in terms of run-time complexity ($O(n)$ instead of $O(n^2)$) and in terms of the experiments we have performed (by a factor of 10 to 100).

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. 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)
8. Bisson, G.: Why and how to define a similarity measure for object based representation systems. Towards Very Large Knowledge Bases (1995) 236–246
9. Maedche, A., Staab, S.: Measuring similarity between ontologies. In: Proceedings of the European Conference on Knowledge Acquisition and Management (EKAW), Springer (2002)
10. Levenshtein, I.V.: Binary codes capable of correcting deletions, insertions, and reversals. Cybernetics and Control Theory (1966)
11. Cox, T., Cox, M.: Multidimensional Scaling. Chapman and Hall (1994)
12. 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
13. 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)
14. 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)
15. Rodrguez, M.A., Egenhofer, M.J.: Determining semantic similarity among entity classes from different ontologies. IEEE Transactions on Knowledge and Data Engineering (2000)
16. Mitra, P., Wiederhold, G., Kersten, M.: A graph-oriented model for articulation of ontology interdependencies. Lecture Notes in Computer Science **1777** (2000) 86+
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