Domain-aware Ontology Matching

Kristian Slabbekoorn[1], Laura Hollink[2,3], and Geert-Jan Houben[2]

[1] Department of Computer Science, Tokyo Institute of Technology, Japan
[2] Web Information Systems Group, Delft University of Technology, The Netherlands
[3] Knowledge Representation and Reasoning Group, VU Amsterdam, The Netherlands

Abstract. The inherent heterogeneity of datasets on the Semantic Web has created a need to interlink them, and several tools have emerged that automate this task. In this paper we are interested in what happens if we enrich these matching tools with knowledge of the domain of the ontologies. We explore how to express the notion of a domain in terms usable for an ontology matching tool, and we examine various methods to decide what constitutes the domain of a given dataset. We show how we can use this in a matching tool, and study the effect of domain knowledge on the quality of the alignment.
We perform evaluations for two scenarios: Last.fm artists and UMLS medical terms. To quantify the added value of domain knowledge, we compare our domain-aware matching approach to a standard approach based on a manually created reference alignment. The results indicate that the proposed domain-aware approach indeed outperforms the standard approach, with a large effect on ambiguous concepts but a much smaller effect on unambiguous concepts.

1 Introduction

The rise of the Semantic Web and Linked Open Data has led to a large number of heterogeneous datasets and ontologies published on the Web [2]. This kind of heterogeneity has created a need to interlink these datasets, i.e. to make explicit that two resources of different datasets represent the same concept. However, the discovery of such correspondences between entities of different ontologies or datasets is not trivial – it is in fact one of the major challenges of the Semantic Web [1].

A large number of ontology matching tools and methods have emerged in recent years [4]. In this paper we are interested in what happens if we take these existing tools and enrich them with knowledge of the domain of a given domain-specific ontology. This is especially relevant in situations where we want to match such an ontology to a more general ontology; a scenario that is often seen on the Linked Data Web. If our source dataset is, for example, a medical vocabulary, it would help us a lot to know which concepts in the target dataset are within the relevant domain of medicine, as disambiguation of concepts is not trivial. In this paper we examine various methods to decide what constitutes the domain of a given dataset. We explore how to express the notion of a domain in terms usable for an ontology matching tool and we study the effect of domain knowledge on the quality of the alignment. The main contribution is the development and evaluation of a domain-aware ontology matching approach geared at interlinking domain-specific ontologies with cross-domain ontologies.

We present an approach that, in a fully unsupervised way, discovers the domain of a domain-specific source ontology expressed in terms of the schema information of the (cross-domain) target ontology. This ensures that matching can be performed by restricting matches to entities within this domain. The approach consists of five phases: (1) a *high-confidence matching phase* in which we determine a small set of mappings between source and target ontology for which we are almost sure they are correct; (2) a *class/category collection phase* in which we collect the schema information of the mapped instances of the target ontology; (3) a *domain pruning phase* in which we translate the domain knowledge into a filter; (4) a *domain optimization phase* in which we optimize this filter; and finally (5) a *domain-aware matching phase* where the actual matching is performed – the domain filter is used in an ontology matching tool to ensure that concepts are only matched if they are within the relevant domain.

We evaluate our approach in two scenarios: we match (1) artists mined from Last.fm in the EventMedia dataset of cultural events[4] and (2) a part of the medical vocabulary UMLS to DBpedia. We have chosen DBpedia because this is the de facto cross-domain centerpiece of the Linked Open Data cloud. This makes it an attractive choice as it contains many concepts (over 3.64 million on September 2011) across various domains, presenting us with the real problems of ambiguous concepts and thus irrelevant candidate matches. By focusing on a single target ontology, we aim to achieve a matching approach that yields high accuracy for the generated links. In addition, it allows us to optimally make use of the schema information that is specific to the target ontology.

In both scenarios, we experiment with different methods to decide what constitutes the domain of a dataset. We use each method in combination with DBpedia Spotlight, a dedicated DBpedia matching tool, to create an alignment. To quantify the effect of domain knowledge in general, and each of the domain-derivation methods specifically, we compare each alignment to a manually created reference alignment. An early version of our approach, with a more limited domain representation and without a domain optimization phase, was evaluated on one of these scenarios and has been published previously [20].

The remainder of this paper is organized as follows. We start by giving a description of the task we are going to perform in section 2. Next, we explain the domain-aware approach, in section 3. Section 4 contains the evaluation of the approach and a discussion of the results. In section 5 we present a discussion of related work, expanding on works dealing with similar approaches to ontology matching. The final section concludes the paper and describes possible future work.

2 Task Description

Domain representation in DBpedia As mentioned, we have chosen DBpedia as the target ontology of our approach, which allows us to use the schema information of DBpedia in the representation of a domain filter. This goes further than built in schema constructs in standard Semantic Web languages: to represent a domain we use *DBpedia classes*, *Wikipedia categories*, *YAGO classes* and *Freebase classes*. The first three types

[4] `http://thedatahub.org/dataset/event-media`

are already available in DBpedia; Freebase classes are derived through Freebase interlinks. Therefore, what we essentially want to find out is how the domain of the source ontology can be mapped effectively to a set of of classes and categories in DBpedia.

Given the full union set of all classes and categories C, a *domain* D can be defined as a subset of C, i.e. $D \subseteq C$. Once we have derived a target domain, we can use it as a filter for the traditional matching process done by (complex) string comparison. The reason we derive a set of classes/categories rather than a single class is that there is usually no perfect mapping to a single target class. Also, the classes contained in DBpedia are either user-generated, or derived from a user-generated base. Therefore, since not all resources have proper classes or categories assigned to them, some redundancy in the domain encapsulation helps to deal with this messiness of data (figure 1).

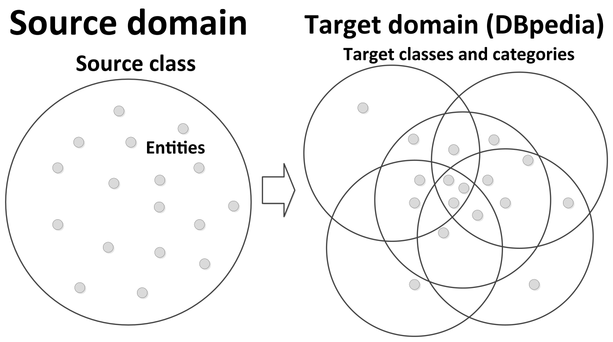

Fig. 1: Mapping from source to target domain.

Source datasets and quality measures We evaluate the approach with two domain-specific ontologies: Last.fm artists and UMLS medical concepts. The Last.fm Artist dataset consists of roughly 50,000 entities. For the UMLS concepts we consider two separate classes of terms: pathologic terms and physiologic terms, which consist of roughly 70,000 resp. 55,000 terms. To illustrate the approach more clearly, we take the Last.fm Artist dataset as running example. Note that these source datasets are used for evaluation purposes only; our approach is independent of the source dataset.

Manual reference alignments R are created for all three datasets. For Last.fm Artists we manually determine links for 1000 randomly selected artists; for each type of UMLS terms we create links for 500 random entities. The links are determined by one person (the first author) – we feel that this suffices, as the reference alignments are used to compare variations of our method (and a baseline), not to give an authoritative performance score to an alignment method. We measure the quality of alignments in terms of F-measure, which is the harmonic mean of recall and precision. Our task is to produce alignments with a high F-score.

Baseline matching tool: DBpedia Spotlight As a starting point for the actual matching task we take DBpedia Spotlight [14], a powerful, off-the-shelf tool for matching textual resources to DBpedia. DBpedia Spotlight has been shown to be able to compete with established annotation systems while remaining largely configurable [14]. Its configurability allows us to include various forms of domain knowledge, and test the effect on the resulting matches.

Spotlight works by first finding *surface forms* in the text that could be mentions of DBpedia resources (the "spotting" function), then disambiguating these surface forms to link to the right DBpedia resources based on context similarity measures (the "disambiguation" function). Its results can be directed towards high precision or high recall by setting two parameters: a "support" threshold for minimum popularity of the associated Wikipedia page (i.e. the number of inlinks to this page from other Wikipedia pages) and a "confidence" threshold for minimum similarity between source text and context associated with DBpedia surface forms.

For our work we do not employ the full range of Spotlight's features – the tool mostly specializes in the annotation of free text, while we deal with a very specific case of text annotation, namely that of entity labels. In the majority of cases it suffices to simply try to disambiguate this full label to find a corresponding DBpedia resource.

3 The Domain-Aware Matching Approach

Figure 2 shows a high-level schematic of the pipeline employed. As input we take an ontology or dataset, and the label(s) and context of each entity. Given the example of the Last.fm dataset, we want to match artists, so we give as input the `foaf:Agent` class of the EventMedia RDF dataset, and as label and context properties `rdfs:label` and `dc:description` respectively.

The approach is then divided into five phases. In the *high-confidence matching phase*, we collect an initial sample of links of which we can be fairly sure they are correct. For each DBpedia resource in this sample of links, classes and categories are collected in the *class/category collection phase*. This collection is pruned to a processable size in the *domain pruning phase*, then optimized to form an adequate domain filter in the *domain optimization phase*. Lastly, the domain-restricted matching is performed in the *domain-aware matching phase*. Each step will be explained in detail in the following sections.

3.1 High-confidence matching phase

Initially, we assume to have no knowledge about our input dataset. The selection of the domain filter is bootstrapped by first attempting to match instances of the source dataset to DBpedia resources without any knowledge of the domain, to obtain an initial linkset from which we want to derive domain information. Ideally, we want to maximize precision for these links while still obtaining a sufficiently large sample to make meaningful

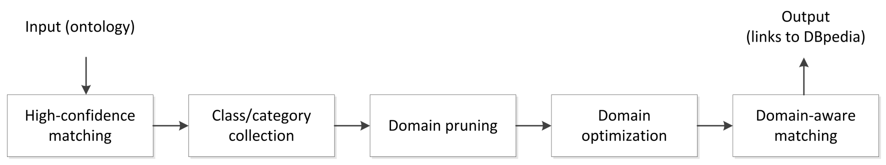

Fig. 2: High-level pipeline view for the domain-aware approach.

generalizations for the full dataset. This is accomplished by applying DBpedia Spotlight without any domain restriction and with parameters set towards high precision, thereby relying solely on the lexicographical similarity of contexts between source and target entities for disambiguation (entity labels and descriptions are compared to the textual content of candidate DBpedia resources). Entities to match are selected at random from the source dataset. In this way, we can generate an initial set of high-confidence links from our data to DBpedia resources. This set of links is used for two separate purposes: (1) the collection of classes and categories for the derivation of a domain; and (2) the optimization of the target domain filter (as discussed in section 3.4).

For the first goal, we want a linkset that is large enough to make accurate predictions about the domain. For the second goal, we want a set of links that resembles a reference alignment, consisting of *positive matches* and *negative matches*. We call the set of high-confidence matches the *high-confidence linkset*.

Definition 1 (Positive match). *A mapping from a source entity to a target DBpedia resource. A single source entity may have multiple positive matches to (distinct) DBpedia resources.*

Definition 2 (Negative match). *A null-mapping for a source entity, i.e. this entity is determined to have no corresponding DBpedia resource.*

Definition 3 (High-confidence linkset). *A* high-confidence linkset R' *is the set of positive and negative matches obtained by bootstrapping the source dataset.*

For the purposes of this experiment, the high-confidence linkset was made to be disjoint with the reference alignment R. The variable name R' was chosen as such because the high-confidence linkset can be regarded as a *virtual* version of a real reference alignment R. For evaluation purposes, we fix the number of matches in R' to 1000. We have experimented with high-confidence linksets containing only positive matches, and linksets containing both positive and negative matches in various ratios. We found that the best results were obtained with a high-confidence linkset containing positive and negative matches in a 500/500 ratio ($R'_{500/500}$). For the sake of brevity, we omit these experiments from this paper and take $R'_{500/500}$ as high-confidence linkset type.

The best value to choose for "confidence," the threshold parameter for context similarity in Spotlight, is somewhat dependent on the dataset. The positive and negative matches are gathered with two *separate* confidence thresholds: to be maximally confident in positive matches, the threshold should be high, i.e. high context similarity and/or low ambiguity are demanded; to be maximally confident in negative matches, the threshold should be low, i.e. low context similarity and/or high ambiguity are demanded. We aim for the strictest thresholds where enough matches are still obtained.

The resulting high-confidence linkset allows us to derive and optimize a domain filter without the use of a manually created reference alignment.

3.2 Class/category collection phase

From the resulting DBpedia resources in the high-confidence linkset we gather the associated classes and categories so that we can later use these as a domain knowledge filter for a matching tool. We gather *DBpedia classes*, *Wikipedia categories*, *YAGO classes* and *Freebase classes*. The first three types are already available in DBpedia; Freebase classes are derived through Freebase links in DBpedia. We use the following collection of classes/categories as a basis for the derivation of an effective domain filter:

1. *DBpedia classes.* These are derived from the manually assigned infobox types on Wikipedia pages. The DBpedia classes are ordered in a shallow, strict hierarchy (i.e. each class has no more than one super-class), so we use the transitivity of the subclass hierarchy and include not only classes directly associated to the resource, but also all super-classes up to the root of the hierarchy.
2. *Wikipedia categories.* These are the categories manually assigned to Wikipedia pages. They are ordered in a broad and non-strict hierarchy (i.e. categories can have multiple super-categories). Therefore we gather only up to 4 ancestors of each, as due to the size and messy structure of the category hierarchy the set would quickly become too large and broad to be useful.
3. *YAGO classes.* These are automatically derived from the Wikipedia category system using WordNet [15][23]. YAGO classes are ordered in a deep, near-strict hierarchy (a select few classes have multiple super-classes), so we gather all super-classes up to the root of the hierarchy.
4. *Freebase classes.* These are taken from the Freebase concept associated with a DBpedia resource (if available). Freebase classes are ordered in a shallow, strict hierarchy of only two levels with no single root node, so all we gather for each concept is the class and its super-class.

3.3 Domain pruning phase

We prune the collection of classes and categories in order to get a tighter encapsulation for the domain. Classes that only occur once or twice over the entire class/category set are typically better left out, as these are likely to be outliers to the domain of interest. Even if not, they tend to be too narrow to play any role. In addition, we need to discard classes that are too broad and cover far more entities than what we want it to do. This is especially necessary as we collect DBpedia and YAGO classes up to the root of the hierarchy, thus including the root classes Thing and Entity.

We develop two *pruning strategies* that can be applied to somewhat limit the initial domain, all the while taking care not to discard potentially relevant domain information.

Low-occurrence domain pruning (LDP) We can get rid of classes/categories that occur too little as follows. We filter out all DBpedia and YAGO classes that occur less than r_d% resp. less than r_y% of the total number of classes found. A percentage is required since we do not know beforehand exactly how many of the top t most occurring classes are too generic, as this depends on the domain of the source dataset. For categories, we do not necessarily need to filter by a percentage since we do not gather super-categories up to the root. Since we only collect them within a specified semantic distance, the very categories with the most occurrences should be a relatively accurate representation of our domain. Therefore we simply select the top t_c categories that occur the most. For Freebase classes, there are only two levels in the hierarchy, hence this effect does not occur here either: we select simply the top t_f classes.

High-generality domain pruning (HDP) A second strategy is needed to eliminate excessive redundancy in the list of classes and categories, i.e. to avoid including a super-class plus all of its sub-classes. We address this by filtering out all super-classes where the sum of the numbers of occurrence of their *direct* sub-classes is greater than $p\%$ of the number of occurrence of the super-class. In other words, for every class (or category) *Super*, if

$$\sum_{i=1}^{n} occs(Sub_i) > \frac{p}{100} occs(Super), \quad (1)$$

where n is the number of direct sub-classes Sub_i of *Super*, then *Super* is removed from the collection. This procedure is dependent on the values chosen for r_d, r_y, t_c and t_f in the LDP step.

See figure 3 for an illustrated example of pruning Last.fm artists with $r_y = 1\%$ (cutoff of 14 occurrences in this case) and $p = 80\%$ for the YAGO hierarchy. Class occurrence numbers are shown in parentheses. The percentages are the proportions of direct sub-class occurrences to this class' occurrences. Classes grayed out were discarded in the LDP step; classes marked red are discarded in the HDP step. Classes marked green represent the final domain. As can be seen, most generic and unrelated classes are successfully filtered out. However, some broad classes, such as "LivingPeople," still remain, which suggests that we need extra measures in order to further refine the domain encapsulation.

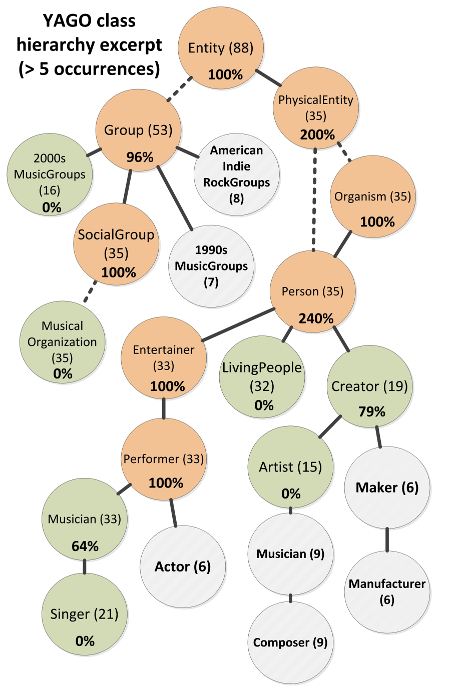

Fig. 3: Top part of the YAGO hierarchy after pruning with $r_y = 1\%$ and $p = 80\%$.

3.4 Domain optimization phase

After pruning classes/categories that are too general or too specific, we further optimize the set of classes/categories in the domain filter by testing which set gives the highest alignment performance. We apply and compare three methods. The first is a metaheuristic-based approach [13], i.e. a generic optimization method that is known to be able to discover (near-)optimal solutions independent of the nature of the optimization problem itself. For this we will apply a genetic algorithm to our problem. The second and third are problem-specific heuristic-based approaches that exploit the structure of our specific problem domain.

Each optimization approach needs a way to evaluate the candidate solutions. As said, we aim to be independent of a manually created reference alignment and instead use the automatically created high-confidence linkset. This approach is only sensible, however, under the assumption that there exists a correlation between performance on the automatically generated high-confidence linkset R' and performance on the manually constructed reference alignment R. Before we go into the details of the three optimization approaches, we first show that this assumption is valid.

Testing the validity of the bootstrap linkset We test various configurations of a domain-aware matcher by comparing the alignments that it creates to the manually created reference alignment and to the high-confidence linkset. Performance on the reference alignment is measured with the F-score. Performance on the high-confidence linkset is measured in the same way, but we call this measure the "virtual" *F'-score* to distinguish it from the "real" F-score.

We evaluate by comparing F'-scores obtained using R' and F-scores obtained using R. We generate 36 different domain filters by first applying the high-confidence matching and domain pruning phases of sections 3.1 and 3.3 with varying settings, and then randomly excluding half of the classes/categories. Each of these 36 domain filters is used in combination with DBpedia Spotlight to produce two alignments: one on the concepts in R' and one on those in R. F- or F'-scores of each alignment are calculated by comparison to R and R' respectively. Figure 4 shows the performance of the 36 domain filters on R and R'. We observe that our assumption seems valid: there is a high correlation between performance on R and R'. To quantify this observation, we repeat the process 15 times. Table 1 shows the mean and standard deviation of the correlations between performance on R' and R, for the Last.fm and UMLS datasets.

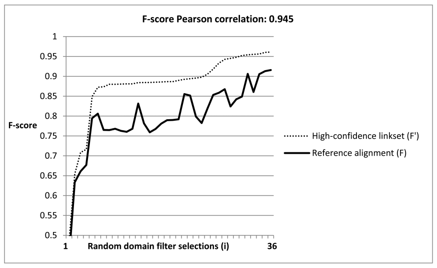

Fig. 4: Correlation between R' and R for an example run (Artists). Values have been sorted by F'_i in ascending order.

	Last.fm	UMLS
$\bar{r}$	0.978	0.980
sd	0.009	0.007

Table 1: Mean and standard deviation of the Pearson correlation $\bar{r}$ between performance on R' and R.

Metaheuristic: genetic algorithm We adapt our problem to a format that can be processed by a genetic algorithm. First, low-occurrence pruning (LDP) is applied to limit the search space to a processable size. The resulting domain D is then encoded as a chromosome, i.e. a bit array, where each bit represents whether the corresponding class/category is included or not. As fitness function to check the quality of a chromosome (domain filter) D, we generate a domain-restricted alignment for the source entities contained in high-confidence linkset R' and calculate F' with regard to R'.

Since discovering the best starting values is known to be very hard [3], we decide to stick with a starting population of 100 chromosomes, and the default values for selection, crossover and mutation that came with the out-of-the-box genetic algorithm implementation employed for this experiment[5]. The algorithm is run for a relatively high number of 50 generations, so as to provide the algorithm ample time to converge to an optimum (we are not concerned with running times considering our type of matching is a one-time process). The fittest remaining solution is chosen as the result.

Problem-specific heuristic: broadness-based domain optimization (BDO) After applying both LDP and HDP on the initial domain selection as described in section 3.3, we can try to optimize what remains of the domain selection D by selectively removing those classes that have a high probability of being too broad. One by one, a class or category is removed from D, and each time the performance F' is calculated. If performance improves, this class/category is left out. Else, it is included in D again. To maximize the effectiveness of this strategy, the order in which we remove classes/categories should be from broad to narrow. Hence D is first sorted by broadness in descending order. The size of a class, i.e. the number of instances that belong to this class (including its descendants), is taken as a measure of its broadness.

We repeat this algorithm for different settings for r_d, r_y, t_c and t_f of the HDP step, keeping track of the domain selection corresponding to the highest F'-score seen so far. The resulting D giving the highest F'-score according to R' is chosen.

Problem-specific heuristic: precision-based domain optimization (PDO) For this approach, we do not apply HDP, but take the resulting D from the LDP step as input. The next step is to test for each individual class/category in D how many entities are contained. This is done by generating alignment instances for the source entities of R' restricted to single-class domain filters, i.e. $|D_{single}| = 1$. Each class/category in D is tested, and for each the *precision* score is stored. Generally, the higher the precision score, the less false positive matches we find and the more relevant the class/category is to our domain of interest. In other words, the higher a precision score, the higher the probability that the class under test is specific to our domain of interest (e.g. restricting to a class "British2000sMusicalGroups" may give a high precision score when the domain of interest is "musical artists"), and vice versa. Therefore we can optimize the domain selection by applying an F'-based hillclimbing on the list of classes sorted by precision score in descending order. First, we put the class with the highest precision in domain filter D_{test} and calculate F'-score according to R'. Then we add the second class down the list to D_{test}. We check F' again, keeping track of F'_{max}, i.e. the maximum F'-score seen, and its associated domain filter D_{max}. F'_{max} and D_{max} are updated every time that $F'_{current} \geq F'_{max}$: even if F'_{max} remains the same, we prefer the largest domain filter. Once we have gone through the entire list, ending with $D_{test} = D$, the current D_{max} is our final domain filter that we take to the matching phase.

3.5 Domain-aware matching phase

Once we have derived a suitable domain filter, the actual matching is performed. We make use of the functionality provided by DBpedia Spotlight. With the assumption

[5] `http://jgap.sourceforge.net/`

that there is now a filter that tightly encapsulates the domain of the source entities to a domain in DBpedia, the context similarity and ambiguity requirements can be loosened, since disambiguation is now largely performed by this filter. We attempt to match the full source entity label(s) to a candidate in DBpedia that fits the domain filter.

4 Evaluation

Evaluation of ontology matching approaches is typically done by comparison to a baseline approach or to other matching systems that perform the same task. Since we are interested in the added value of using domain knowledge, the evaluation will primarily be done with regard to a baseline approach. We compare all domain optimization methods to each other and to two baselines that do not use domain information. Performance is evaluated by comparison to manually created reference alignments to obtain precision, recall and F-scores. All experiments are performed three-fold – once on the Last.fm dataset, and once for the Pathologic and Physiologic Function classes in UMLS.

4.1 Evaluation strategy

In this section we list all approaches we test. We divide the approaches into three categories: baselines, basic strategies and the heuristic approaches described in section 3.4.

Baselines As a baseline, we use DBpedia Spotlight without considering a domain filter at all. Such a baseline is essential in showing the added value of a domain-aware matching approach. However, depending on the dataset, the optimal settings of Spotlight might vary. For the purpose of this evaluation, aside from a *true baseline*, we will also artificially derive an *optimized baseline*. For the true baseline, Spotlight's settings are kept consistent with those of the the domain-aware approaches, meaning we do not rely on similarity thresholds. For the optimized baseline, Spotlight's matching parameters are optimized based on F-score according to R. This provides a theoretical upper-bound on performance for matching with Spotlight without a domain filter.

Basic strategies Two light-weight, basic strategies are discussed that may obtain a good result for relatively little computation time. We introduce an *optimized random selection* strategy, and an *optimized pruning* strategy – the latter applies the pruning strategies of section 3.3 in a way that we can optimize the result.

Optimized random selection A very basic optimization approach is to take the best result from a random sample. The low-occurrence domain pruning (LDP) strategy is executed to obtain an initial domain selection. From this domain, we generate 100 subselections randomly by including/excluding each class/category with a probability of $\frac{1}{2}$. We calculate the F'-score of each and choose the best performer as the domain filter.

Optimized pruning We apply both the LDP and high-generality domain pruning (HDP) strategies with different parameter settings, and choose as domain filter the result that provides the highest F'-score. This relatively cheap approach effectively deals with the high number of parameters of the pruning strategies.

Heuristics-based approaches In section 3.4, we introduced one metaheuristic, the *genetic algorithm*, and two problem-specific heuristics, broadness-based domain optimization (*BDO*) and precision-based domain optimization (*PDO*), that we can apply to the derivation of a domain filter. The genetic algorithm and PDO are applied just as described in that section. The BDO approach is applied as an additional step to the optimized pruning basic strategy described before.

4.2 Experimental results

The main results for all methods and scenarios are summarized and listed in this section. F' results are included because they provide insight into how each optimization method works. As explained before, while the correlation between F' and F has been shown to be very high, it is not a *perfect* correlation, so an increase in F' does not necessarily translate to an increase in F. Aside from the matching scores, we also list computation complexity in terms of the (estimated) number of alignments for the (1000) entities in R' we need to generate on average for each approach.

Last.fm Artist

Approach	F'	*Precision*	*Recall*	F	Complexity
PDO	0.884	0.964	0.889	0.925	~ 70
Genetic algorithm	0.889	0.960	0.868	0.912	8600
Optimized pruning	0.864	0.935	0.879	0.906	108
BDO	0.890	0.965	0.837	0.897	~ 1080
Optimized baseline	n/a	0.798	0.798	0.798	n/a
Random selection	0.768	0.722	0.865	0.787	100
True baseline	n/a	0.673	0.857	0.754	n/a

Table 2: Results for each approach on the Last.fm artist dataset, sorted by F-score.

The resulting virtual F-score F' and actual precision *Precision*, recall *Recall* and F-scores F for each approach on the Last.fm dataset are displayed in table 2. Note that for complexity, there may be some variability depending on the chosen parameters, in which case an estimation is given, denoted by "$\sim$". The value for the genetic algorithm is based on the selection, crossover and mutation rate per generation, for 50 generations.

For Last.fm, a clear improvement can be seen over the baseline for any of the domain filter-based approaches barring a random selection. The best performing approach is the precision-based optimization, giving an F-score of 0.925, which is $0.925 - 0.798 = 0.127$ (12.7%) higher than the optimized baseline and $0.925 - 0.754 = 0.171$ (17.1%) higher than the true baseline in terms of absolute F-score. For reference, the domain filter that yielded this best score consists of the following classes/categories: *DBpedia classes:* `Band`. *YAGO classes:* `MusicalOrganization`, `2000sMusicGroups`. *Categories:* `Musicians`, `Musicians_by_nationality`, `Musicians_by_genre`. *Freebase classes:* `music/musical_group`, `music/group_member`, `music/artist`.

Differences amongst the optimization approaches are relatively small – the most significant difference exists between PDO and BDO ($0.925 - 0.897 = 0.025$ (2.8%)).

Additionally, it shows that a higher F' does not always lead to a higher F when differences between F' are small, which suggests that optimization on an automatically created set of links works, but not perfectly.

UMLS Pathologic Function

Approach	F'	*Precision*	*Recall*	F	Complexity
Genetic algorithm	0.686	0.974	0.931	0.952	8600
PDO	0.674	0.964	0.936	0.950	~ 70
Optimized baseline	n/a	0.960	0.941	0.950	n/a
True baseline	n/a	0.950	0.941	0.946	n/a
Random selection	0.670	0.955	0.931	0.943	100
Optimized pruning	0.672	0.994	0.877	0.932	108
BDO	0.679	1.000	0.852	0.920	~ 1080

UMLS Physiologic Function

Approach	F'	*Precision*	*Recall*	F	Complexity
PDO	0.741	0.965	0.925	0.945	~ 70
Genetic algorithm	0.758	0.979	0.891	0.933	8600
BDO	0.758	0.979	0.891	0.933	~ 1080
Optimized pruning	0.753	0.981	0.883	0.930	108
Random selection	0.719	0.902	0.928	0.915	100
Optimized baseline	n/a	0.852	0.975	0.909	n/a
True baseline	n/a	0.837	0.975	0.901	n/a

Table 3: Summarized results for the UMLS datasets.

The summarized results for the UMLS datasets “Pathologic Function” and “Physiologic Function” are displayed in table 3. For Pathologic Function, there is no clear difference between any of the approaches. This can be attributed to the fact that the entities in this class are highly specific terms (e.g. medical terminology for diseases) that are very unambiguous, so that correct matches to DBpedia resources are easily found. Here, we see that a domain filter can potentially even harm performance compared to the baseline. This is because entities that do not have very specific class information in DBpedia still yield a correct match in the baseline approaches due to the unambiguity of the concept name, but may get excluded from a class based domain filter.

For Physiologic Function, there is a clear improvement over the baselines again, although it is less significant than in the Artist case. This dataset contains more ambiguity than the Pathologic Function dataset (e.g. common mental processes such as “Recognition,” and genes with abbreviations that are commonly used to refer to other things), but is still quite specialized, therefore the impact of a domain restriction is not as obvious as for very ambiguous entity labels such as artist and band names.

The overall results show that relying solely on the F'-scores, choosing whichever method provided the highest value, may not always be the best option – the type of approach used seems to play a role as well. The precision-based optimization approach performs quite well, being the best performer on the Artist and Physiologic Function datasets and the second best performer on the Pathologic Function dataset. This despite the fact that it never attains the highest F' compared to the other approaches. It is also the fastest approach, requiring roughly 70 alignment generations.

Statistical significance of the results The precision and recall scores above are based on only a sample of all matches that our approach could produce. To extrapolate sample evaluations to statements about general performance, [6] shows that if p is the true proportion of matches that is correct (which is unknown), n the size of the sample used to approximate p (i.e. the size of the reference alignment) and $\hat{P}$ the approximation of p based on the reference alignment, then this approximation lies in the interval

$$\hat{P} \in [p - \delta, p + \delta] \text{ where } \delta = \frac{1}{\sqrt{n}} \tag{2}$$

with 95% confidence. δ is thus the 95% confidence margin of error for the result. If one result falls within this range of another result, then they do not differ sufficiently from each other to state with certainty that one is better than the other. We apply this calculation on the F-scores for each approach to verify their statistical significance.

For the Last.fm dataset, we have $n = 1000$, so the margin of error δ becomes $\frac{1}{\sqrt{1000}} = 0.032$. We can therefore conclude that all optimization approaches aside from the random selection yield a significant improvement over the baselines – the smallest difference is between the optimized baseline and the broadness-based approach, which is $0.897 - 0.798 = 0.099$. Differences among optimization approaches are at most 0.028 (between precision-based and broadness-based), so we cannot state that one optimization approach is better than the other with 95% confidence.

For the UMLS classes, we are dealing with reference alignments of size 500 for both, so here the 95% confidence margin of error for both is $\frac{1}{\sqrt{500}} = 0.045$. For Pathologic Function, none of the optimization approaches are significantly better than the baselines. For Physiologic Function, the largest difference is between the true baseline and precision-based optimization, which is a difference of 0.044. This would be borderline insignificant if we considered this as an individual occurrence, but we can see that *each* optimization approach in fact outperforms the baseline by at least 0.029. Since the optimization approaches consistently perform better, this suggests that the improvements over the baseline are significant.

5 Related work

There has been a great amount of research done into the topic of ontology matching from the perspective of a variety of fields, such as linguistics, AI and knowledge management. As a result, a large number of ontology matching tools and methods have emerged in recent years. Surveys on the current state-of-the-art and future challenges of ontology matching have been described in [4][16]. A great majority of these systems focus on the use of string similarity mechanisms in order to determine correspondences between entities of two different ontologies or datasets – the labels of both entities might be compared in a strict or fuzzy way [22], or additional properties such as aliases or synonyms might be inferred from a background ontology such as WordNet [15][5], or translations of labels could be considered [21]. Other than property comparison, one might turn to look at the schema in which entities in both ontologies are ordered, such as semantic classes and super-classes, or relations with other entities within the same ontology [19]. In other words, this type of approach to ontology matching exploits the hierarchy of the ontology in order to determine links between entities. However, it is often tricky to derive these types of correspondences, as there is usually a string similarity matching step involved – in this case between the names of classes.

In this paper, we derived the domain of entities by bootstrapping the matching process – we generated a sample of high-confidence matches to DBpedia, then derived and optimized a domain filter from this sample. We focus on discussing related work that involves either (1) the matching of entities to DBpedia, or the use of DBpedia or other Linked Data sets as auxiliary knowledge base, (2) the incorporation of domain knowledge into the linking process, or (3) the (parameter) optimization of matchers.

The work in [12] matches entities from a database of the BBC to DBpedia by finding candidates in DBpedia based on string-similarity and wiki inlink metrics, then comparing classifications of their own entities to the classes and categories of the DBpedia candidate resources. In our case, we do not make use of classification of our source data, as this is information not always available (this is the case for Last.fm artists).

Several methods have been proposed that leverage knowledge from DBpedia or other Semantic Web sources to derive links between arbitrary Linked Data sets. BLOOMS+ [9][10] is an ontology matching system that uses the Wikipedia category hierarchy to bootstrap the process of finding schema-level links between Linked Data sets. We exploit the Wikipedia category hierarchy in a similar fashion; not to find matches directly but to find categories (and classes) that effectively describe our domain of interest. In [18], a paradigm is proposed for harvesting the Semantic Web to find related background knowledge to assist in matching two ontologies. Rather than relying on a single, high-quality knowledge source, this approach aims to automatically combine multiple heterogeneous sources. Drawbacks of the approach are its use of string comparison to find related ontologies and its disregard of the quality of the ontology selected. While results are promising given the generality of the approach, it does not match up to more focused systems such as ours. However, it can be used to complement existing techniques.

In [24], English Wikipedia pages are matched to their Chinese-language counterparts in the Baidu Baike knowledge base. The authors predict correspondences by three language-independent features: in- and outlink homophily, category homophily and author interest. In essence, a "domain" is determined for each individual page, and the page that best matches this from the target dataset is selected for linking. This method requires a large set of existing links in order to train a model, however, and is therefore quite different from our scenario. Kalyanpur et. al. [11] describe a type coercion framework for use in a question answering scenario; in this case for IBM's Watson [7]. Their system makes extensive use of DBpedia and domain knowledge extracted from DBpedia. For our approach, we obtain candidate DBpedia resources in much the same way, and similarly leverage the class/category hierarchies to improve the quality of matching. The main difference is that the Watson system attempts to find the correct candidate class by string comparison and hierarchy-based alignment with the source (question) type, while we do not assume to have source type information available.

The authors of [8] present an approach to automatically generate linkage rules from a set of reference links. A genetic algorithm is applied to learn which rules and parameters work best on a dataset, based on performance according to a partial reference alignment. They show that the automatically learned rules achieve similar accuracy to those manually created by humans. In [17], ECOMatch is proposed. ECOMatch is a generic parameterization system designed for use on top of any ontology matching system. It automatically determines a suitable parameter configuration based on user-provided example mappings. The authors show that a correlation exists between scores of a configuration according to a portion of a reference alignment and a full reference alignment. Some aspects of our approach are similar to these two works. Like the first work, we also use a genetic algorithm for optimization, but rather than learning linkage rules to find correspondences, we learn a domain filter to restrict correspondences to. ECOMatch is also different to our work in that we do not try to find an optimal configuration for a matcher. Most importantly, unlike both described works, we do not require a human-constructed set of reference links to find the most suitable domain filter.

6 Conclusion

This paper presented work on the development and evaluation of a domain-aware ontology matching approach. We showed that we can improve the quality of generated links over traditional approaches by bootstrapping the matching process – we derive a filter in a fully unsupervised way that describes and delimits the domain of the source dataset in terms of classes and categories collected from DBpedia.

We showed that when the dataset we want to match to DBpedia is particularly ambiguous, such as is the case for Last.fm artists, a significant gain in matching quality can be obtained by first bootstrapping these matches and deriving a domain filter to restrict them. With the best-performing approach, a 17.1% improvement in F-score was gained over a baseline, domain-unaware approach. For the UMLS "Physiologic Function" subdomain, which is a much more specialized dataset but still contains some ambiguous terms, we still saw an improvement of 4.4%. On the other hand, given an extremely specialized and unambiguous dataset such as the "Pathologic Function" subdomain of the UMLS, matching with a domain filter does not provide a significant improvement, and can potentially even hurt quality.

Future work Our method is currently fixed to DBpedia as target ontology, and DBpedia Spotlight as matcher, but could theoretically work in a generic ontology matching system as well. Future work would be to generalize the current system to also support different target ontologies, or to integrate our approach as a separate step into already existing ontology matching systems. This would also allow for evaluation of our method with regard to existing systems.

Secondly, due to this reliance on DBpedia Spotlight – a tool catered to the annotation of free text – certain valid surface forms are not recognized due to being too common. We expect that creating a custom parser for Spotlight that is specialized towards ontology matching could significantly improve the absolute matching scores.

Acknowledgements

This work is partly funded by the FP7 projects EURECA under grant nr. 288048 and ImREAL under grant nr. 257831.

References

1. V. R. Benjamins, J. Contreras, O. Corcho, and A. Gómez-pérez. Six Challenges for the Semantic Web. In *In KR2002 Semantic Web Workshop*, volume 1, 2002.
2. C. Bizer, T. Heath, and T. Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, Mar 2009.
3. A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter Control in Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141, July 1999.
4. J. Euzenat and P. Shvaiko. *Ontology matching*. Springer, 1st edition, July 2007.
5. F. Giunchiglia, M. Yatskevich, and P. Shvaiko. Semantic matching: Algorithms and implementation. Technical report, University of Trento, Jan. 2007.
6. W. V. Hage, A. Isaac, and Z. Aleksovski. Sample evaluation of ontology-matching systems. In *Fifth Int. Workshop on Evaluation of Ontologies and Ontology-based Tools, ISWC 2007*.
7. IBM. Watson, Retrieved on December 14 2011.
8. R. Isele and C. Bizer. Learning linkage rules using genetic programming. In *Proceedings of the Sixth International Workshop on Ontology Matching at ISWC 2011*, page 13, 2011.
9. P. Jain, P. Hitzler, A. P. Sheth, K. Verma, and P. Z. Yeh. Ontology alignment for linked open data. In *Proceedings of the 9th Intl. Semantic Web Conference*, pages 402–417, 2010.
10. P. Jain, P. Yeh, K. Verma, R. Vasquez, M. Damova, P. Hitzler, and A. Sheth. Contextual ontology alignment of lod with an upper ontology: A case study with proton. In *The Semantic Web: Research and Applications*, volume 6643 of *LNCS*, pages 80–92. 2011.
11. A. Kalyanpur, J. W. Murdock, J. Fan, and C. Welty. Leveraging community-built knowledge for type coercion in question answering. In *Proceedings of ISWC'11*, pages 144–156, 2011.
12. G. Kobilarov, T. Scott, Y. Raimond, S. Oliver, C. Sizemore, M. Smethurst, C. Bizer, and R. Lee. Media meets semantic web: How the bbc uses dbpedia and linked data to make connections. In *Semantic Web: Research and Applications*, LNCS 5554, p723-737. 2009.
13. S. Luke. *Essentials of Metaheuristics*. lulu.com, Mar. 2011.
14. P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. Dbpedia spotlight: Shedding light on the web of documents. In *Proc. of the 7th Intl. Conference on Semantic Systems*, 2011.
15. G. A. Miller. WordNet: a lexical database for English. *Commun. ACM*, 38(11):39–41, 1995.
16. S. Pavel and J. Euzenat. Ontology matching: State of the art and future challenges. *IEEE Transactions on Knowledge and Data Engineering*, 99(PrePrints):1, 2012.
17. D. Ritze and H. Paulheim. Towards an automatic parameterization of ontology matching tools based on example mappings. In *Proceedings of the Sixth International Workshop on Ontology Matching at ISWC 2011*, volume 814, page 37. CEUR-WS, 2011.
18. M. Sabou, M. d'Aquin, and E. Motta. Exploring the semantic web as background knowledge for ontology matching. *J. Data Semantics*, 11:156–190, 2008.
19. P. Shvaiko and J. Euzenat. A survey of schema-based matching approaches. *Journal on Data Semantics IV*, 3730:146–171, 2005.
20. K. Slabbekoorn, L. Hollink, and G.-J. Houben. Domain-aware matching of events to dbpedia. In *Proc. of the DeRiVE'11 workshop, ISWC 2011*, volume 779. CEUR-WS, 2011.
21. D. Spohr, L. Hollink, and P. Cimiano. A machine learning approach to multilingual and cross-lingual ontology matching. In *ISWC'11*, volume 7031 of *LNCS*, pages 665–680, 2011.
22. G. Stoilos, G. Stamou, and S. Kollias. A String Metric For Ontology Alignment. In *Proceedings of ISWC'05*, volume 3729 of *LNCS*, pages 624–637, November 2005.
23. F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *Proceedings of WWW'07*, pages 697–706, 2007.
24. Z. Wang, J. Li, Z. Wang, and J. Tang. Cross-lingual knowledge linking across wiki knowledge bases. In *Proceedings of WWW'12*, pages 459–468. ACM, 2012.