Linking Semantic Desktop Data to the Web of Data

Laura Drăgan[1], Renaud Delbru[1], Tudor Groza[2], Siegfried Handschuh[1], and Stefan Decker[1]

[1] Digital Enterprise Research Institute (DERI), National University of Ireland, Galway
firstname.lastname@deri.org, http://www.deri.ie
[2] School of ITEE, The University of Queensland, Australia
firstname.lastname@uq.edu.au, http://www.itee.uq.edu.au

Abstract. The goal of the Semantic Desktop is to enable better organization of the personal information on our computers, by applying semantic technologies on the desktop. However, information on our desktop is often incomplete, as it is based on our subjective view, or limited knowledge about an entity. On the other hand, the Web of Data contains information about virtually everything, generally from multiple sources. Connecting the desktop to the Web of Data would thus enrich and complement desktop information. Bringing in information from the Web of Data automatically would take the burden of searching for information off the user. In addition, connecting the two networks of data opens up the possibility of advanced personal services on the desktop.
Our solution tackles the problems raised above by using a semantic search engine for the Web of Data, such as Sindice, to find and retrieve a relevant subset of entities from the web. We present a matching framework, using a combination of configurable heuristics and rules to compare data graphs, that achieves a high degree of precision in the linking decision. We evaluate our methodology with real-world data; create a gold standard from relevance judgements by experts, and we measure the performance of our system against it. We show that it is possible to automatically link desktop data with web data in an effective way.

Keywords: Semantic Desktop, Semantic Web, Linked Data, Personal Information Management

1 Introduction

The Semantic Desktop aims to enable better organization of the personal information on our computers, by applying semantic technologies on the desktop. Just like Linked Data connects distributed data on the web, creating a network of interlinked information, the Semantic Desktop connects personal data across

application boundaries on the desktop, creating a network of personal information. However, information on our desktop is often incomplete, as it is based on our subjective view, or limited knowledge about an entity.

On the other hand, the Web of Data contains information about virtually everything, generated by multiple sources, and theoretically unlimited. Connecting the desktop to the Web of Data would thus enrich and complement desktop information. Bringing in information from the Web of Data automatically would release the user from the burden of searching for information.

Connecting the two networks of information opens up the possibility of personal services on the desktop which use external data, but in the personal context of the user, highly connected to his personal data and focused on his interests. One such example is a service that finds implicit links between the publications that the user has on the desktop, and provides recommendations to other publications on the same topics, by the same authors, or related in another way. Another desktop service could use information from the Web of Data to notify the user of new concert dates in his area, based on the latest or most popular artists played on the desktop. web data can also be used as a point of reference when working collaboratively, e.g., documents linked by the user to people, projects, or other resources from his semantic desktop can be shared together with the annotations, which can be accessed and reused outside of the semantic desktop where they were generated.

From the perspective of interlinking information, and using the frameworks provided by the Semantic Desktop and the Web of Data, we have separate islands of knowledge, both containing similar data, related to the same topics of interest to the user, but disconnected from each other.

The disconnection appears in two forms:

- The data on the desktop, although similar to that on the Web of Data, is described using specific *desktop ontologies*, which are different from the ones found on the Web of Data. This schema mismatch makes interlinking data from the two datasets difficult.
- Identifiers (URIs) on the desktop are local to the desktop data space, they are not globally unique and cannot be dereferenced as normal Linked Data URIs are. Hence, it is impossible to access and connect to local data from the Web of Data.

To tackle this disconnection, it is necessary to create links between desktop identifiers and web identifiers that refer to the same real-world thing. This means we need to compare the data graph describing the entity on the desktop with the data graph of an entity on the web. Leaving aside the use of different terminology within the data, the Web of Data is large, billions of entities across hundreds of thousands of datasets. From this vast amount of information we must find and retrieve a relevant subset of entities, that are potential candidates with the desktop entity. Then we must decide if the candidates are similar enough with the desktop entity to create a link between the two. Because we wish to make the interlinking automatic, we must be able to decide with a high degree of precision which candidates among this subset are in fact referring to the same entity.

Our solution tackles the problems raised above by using a semantic search engine for the Web of Data, such as Sindice, to find and retrieve a relevant subset of entities from the web. We then present a matching framework, using a combination of configurable heuristics and rules to compare data graphs, that achieves a high degree of precision in the linking decision.

We evaluate our methodology with real-world data. We create a gold standard from relevance judgements by experts, and we measure the performance of our system against it.

Our solution proves that interlinking the two environments is feasible, and even more, it yields good results. Connecting desktop data with the web enables the system to bring web data to the user, instead of the user having to go find it by himself.

The paper is organised as follow. In Section 2, we start by presenting the Nepomuk Semantic Desktop, as it represents the framework on which we base our solution. We continue with the related work section. In Section 3 we describe the process for finding web aliases for desktop resources, and continue in Section 4 with the implementation of the process and a detailed description of the matching algorithm. We describe the set-up of the evaluation we performed and the results in Section 5. We discuss some of the results in Section 6, before concluding.

Throughout the paper we consider that all the data we are working with is represented as RDF — both on the desktop and on the web. When we mention the desktop, we always mean the Semantic Desktop, more specifically the Nepomuk instantiation of the Semantic Desktop. Similarly, when we mention web data, we refer to the Web of Data. In our implementation we only use Web of Data sources, which are freely available online. However, this is not a requirement of the system, since new data sources can be easily plugged in.

2 Background

In this section, we first provide an overview of the Nepomuk Semantic Desktop and the infrastructure it provides. Next, we review existing approaches for entity linking and entity identity management, and finally compare our entity matching framework with Silk, a linking discovery framework for the Semantic Web.

2.1 Semantic Desktop and Nepomuk

The Semantic Desktop aims to solve the problem of information interlinking and to help managing and organising in a better way our personal data by applying Semantic Web technologies on the desktop. The Semantic Desktop is gaining momentum by the adoption and integration of the Nepomuk framework [2] into mainstream desktop environments.

The Nepomuk Semantic Desktop defines and uses a set of ontologies[3], complemented by ontologies defined by the community, like Xesam[4].It also defines an

[3] `http://www.semanticdesktop.org/ontologies/`

[4] `http://xesam.org/main/XesamOntology` - is used in Nepomuk-KDE

extension to RDF called Nepomuk Representational Language (NRL)[5], which adds Named Graphs and Graph Views to RDF/S and introduces the closed world assumption to the data.

The ontologies describe various aspects of desktop use cases for personal information management. The central ontology is the Personal Information Model (PIMO)[6]. According to its specification [13], "PIMO is based on the idea that users have a mental model to categorize their environment", and "each concept in the environment ... is represented as [a] Thing in the model". PIMO defines high level types like Person, Project, Event and Task. The desktop ontologies also include Nepomuk Annotation Ontology (NAO) which allows users to attach tags and ratings to the resources, Nepomuk Contact Ontology (NCO) which describes contact information for people and organizations, Task Model Ontology (TMO) which describes personal tasks and to-dos, etc. All the data is stored in a central repository that is accessible and shared across applications.

2.2 Related Work

The problem of entity linking is well known across various research communities with a variety of different names, such as record linkage [8], entity resolution [1], reference reconciliation [6] or object consolidation [9]. A wide variety of algorithms has been developed for resolving the coreference problem, but record linkage between distributed databases is still considered a difficult problem.

Recent initiatives within the Semantic Web community address the problem of linking entities across data sources. Jaffri et al. describe the phenomenon of proliferation of URIs and propose a Consistent Reference Service to manage URI equivalences [10]. The OKKAM project [4] proposes an infrastructure for assigning global identifiers at web scale. These approaches are more focussed towards the management of entity identity on the web, but do not provide easy means to create new links between data sources. Similar to our approach, Raimond et al. describe an algorithm and its implementation GNAT, for linking a personal music collection to corresponding MusicBrainz resources [11]. The approach measures recursively the similarity of the resource graphs from the two datasets, with the restriction that the same vocabularies are used in both. By contrast, using property paths in our mappings, we eliminate the need for recursion while still propagating the measures from connected resources. Silk is a framework to help linking multiple entities between two datasets [3]. It relies on user-defined rules and various string matching algorithms to measure the similarity between two entities. In this case it is necessary to know a priory which specific dataset to link to and to perform manual configuration of the matching algorithms, something that requires a high degree of expertise. Hogan et al. [9] and Saïs et al. [12] propose logical-based methodologies for merging identifiers of equivalent entities across multiples knowledge sources. While being precise,

[5] `http://www.semanticdesktop.org/ontologies/nrl`
[6] `http://www.semanticdesktop.org/ontologies/pimo/`

these techniques do not have a very good recall and are demanding in term of computation.

The most relevant approach related to ours is the Silk framework. We provide a generic matching process that the user can configure based on its own expertise in order to get more precise results. However, our approach differs by the fact that the matching process is not restricted to link data between two predefined information sources. On the contrary, our approach gives the possibility to link desktop data with an arbitrary number of external data sources. This makes the problem harder since we are generally unaware of the data structure or schema of these data sources. We therefore need to first find potential entities of interest among a vast number of data sources, then retrieve a partial description of these entities and rely on more complex entity matching algorithms. This first step can be seen as a blocking pass [7] to reduce the information space before executing complex matching algorithms. The blocking step is implemented on top of the boolean query model for centralized search systems such as Sindice [14] and on top of the SPARQL query language for specific data sources providing a SPARQL endpoint.

3 The Process of Finding Web Aliases

The goal of the algorithm and system is to find web aliases for desktop resources. A web alias is a web entity identifier, i.e., URIs, that represents the same real-world thing as the desktop entity to which it was matched. To find web aliases, we use the information available on the desktop, like the contact information from the address book for people, or metadata of music files for songs, albums and artists. We also make use of knowledge about the desktop ontologies and the way data is organized and used on the desktop. The desktop data is used throughout the process, which consists of several steps:

1. Candidate Selection
   - Query and identify candidate entity URIs from various Web of Data sources
   - Retrieve data for each of the candidate from the appropriate Web of Data source.
2. Candidate Filtering
   - Compute similarity score based on the data of the entities.
   - Filter the candidates based on the similarity score.

The first step requires identifying a list of candidate entities and obtaining the data available about them. There are several options to do this: (i) through a small set of sources that we know have the data we need, and querying each of them independently for possible candidates, or (ii) through a search engine for the Web of Data, like Sindice [14], which indexes millions of documents containing semantic mark-ups. Each option has use cases where it is more suitable than the other. Querying specific sources is preferred for instance, if the desktop

data we want to find aliases for is from a very specific domain, like cancer research, or when we are interested only in results from an organization's internal repository. Using a search engine is best when the information sources to query are not known a priori. It also has the advantage of covering a large number of information sources with only one query, and of selecting the most relevant data sources and candidates with respect to the query via the search engine ranking system. However, in the case of ambiguous entities, the latter option has the disadvantage of returning too many unrelated results, thus making the entity selection more difficult.

Once a list of candidates is available, we compute a similarity score for each of them with respect to the desktop entity. The algorithm checks first if the types of the candidate entities correspond to the type of the desktop entity, and discards the ones that do not. Only then, the data of the entities are examined and the properties and corresponding values are compared. If required, the algorithm looks at other related entities and their properties. The values of the properties are compared using either exact string matching or string similarity techniques.

4 Implementing the Process

We implemented the process described above, in a desktop daemon that finds web aliases for desktop entities. It sequentially searches for aliases for all resources that have no alias listed, and for the resources that changed since the last time aliases were determined for them. In the case when a resource is revisited, the previously found aliases are discarded and new ones are determined.

New links are created on the desktop between the local and the web resources, once the aliases are found. They can be used to enhance the available desktop data about the entities, or as entry points to access further information about them online.

The tool has two major components, each handling one step of the matching process. A query component that initiates the search and identifies the candidates, and a matching component that filters the candidates based on similarity measures.

4.1 The Query Modules

The query component can use either generic search engines or specific data sources. Therefore, we chose to make the query component plugin-based, thus allowing various new sources to be connected if needed. The query modules are responsible for finding the initial list of candidates, as well as for retrieving the data for each candidate. The maximum number of candidates to retrieve from a data source can be set as a parameter in the configuration. We allow three types of plugins:

SWSE — connect to semantic search engines, through their APIs. We provide a plugin of this type for Sindice.

Sparql — connect to sources that provide a SPARQL endpoint. We provide plugins of this type for DBpedia and the Semantic Web Conference Server.

Custom — connect to other sources, possibly ones that do not expose any data as RDF (e.g., relational databases or third-party APIs like last.fm).

Both DBpedia and SWC are indexed by Sindice, therefore the Sindice plugin is the only one enabled by default.

In the Sindice module, the initial query, which determines the list of candidates, is constructed using all the value properties of the desktop entity, combined using the boolean conjunction operator "OR". Multiple word terms are also tokenised and the tokens are added to the query. We rely on the search engine to interpret the query and rank higher the results that match most of the terms. For the music album shown in Figure 3, the query constructed is:

Example 1. "Bee Gees" OR "One Night Only" OR "1998" OR "Bee" OR "Gees" OR "One" OR "Night" OR "Only"

4.2 The Matching Module

The matching module computes a similarity score for each pair (*desktop entity—web candidate entity*). The way the score is computed depends on a set of parameters:

String matching (SM) — If this parameter is set to `true`, the matching module will use string similarity measures where appropriate. Currently the system supports Monge Elkan and Chapman distances. If the value is set to `false`, the matching module uses exact matching of property values.

Weighted properties (WP) — If `true`, the matching module will use weights for the properties compared, otherwise, all properties contribute the same to the final score.

Multi-valued properties (MVP) — If `true`, properties that have more than one matching value will contribute to the score proportionally to the number of values.

The algorithm also uses a set of mappings from the desktop ontologies to some of the more popular web vocabularies, like FOAF. There are two kinds: type mappings (see Figure 1 for an example) and property mappings, each described in a separate file. The property mapping supports paths of properties. For example, you can express a path composed of the property `dbpedia:artist` and `foaf:name` as shown in Figure 2. The mappings are relatively static configurations of the system. We have created a set of mappings for the most common ontologies, which can be used out of the box by the end users. Power users can edit the mapping files according to their need.

The algorithm for computing the score works as follows. Considering e_d and e_w the pair of entities to be compared, it first determines the sets T_{e_d} and T_{e_w} of types for each entity, and the set $Map[T_{e_d}]$ of types to which the elements of T_{e_d} are mapped to. If no types are matching, i.e., $T_{e_w} \cap Map[T_{e_d}] = \phi$, it gives a

```
"http://www.semanticdesktop.org/ontologies/2007/11/01/pimo#Person":{
   "mapping":[
      "http://xmlns.com/foaf/0.1/Person",
      "http://xmlns.com/foaf/0.1/Agent",
      "http://dbpedia.org/ontology/Person",
      "http://www.w3.org/2000/10/swap/pim/contact#Person"
      "http://rdf.data-vocabulary.org#Person" ]}
```

Fig. 1. Type mapping for pimo:Person.

```
"http://www.semantcdesktop.org/ontologies/2009/02/19/nmm#performer##
    http://www.semanticdesktop.org/ontologies/2007/03/22/nco#fullname":{
  "mapping":[
    "http://dbpedia.org/property/artist",
    "http://xmlns.com/foaf/0.1/maker##http://xmlns.com/foaf/0.1/name",
    "http://dbpedia.org/ontology/artist##http://xmlns.com/foaf/0.1/name"
  ],
  "approx":"true",
  "thresholds":[
    "MongeElkan:0.7",
    "Chapman:0.8"
  ],
  "weight":"0.7" }
```

Fig. 2. Property mapping for nmm:performer.

score $score(e_w) = 0$, and stop the matching. Otherwise, it continues the process by evaluating the properties.

The evaluation of the properties is driven by the relations and properties of the desktop entity e_d. For each property p_{e_d}, the algorithm retrieves the list of values $V(p_{e_d}) = \{v \ : \ \{e_d \ p_{e_d} \ v\}\}$. Based on the list of property mappings $Map[p_{e_d}]$, it determines the set of values $V(p_{e_w} \cap Map[p_{e_d}])$ that the properties from $Map[p_{e_d}]$ have in common with e_w. If there is no value in common, i.e., $V(p_{e_d}) = \phi$ or $V(p_{e_w} \cap Map[p_{e_d}]) = \phi$, it skips the pair and there is nothing added to the score. Otherwise, it continues the process by measuring the similarity between values.

The evaluation of values is performed using string similarity between each pair of values $(v_d, v_w) \in V(p_{e_d}) \times V(p_{e_w} \cap Map[p_{e_d}])$. The algorithm creates a sparse matrix where the value of a cell contains a string similarity score between 0 and 1. Let $sum_{p_{e_d}}$ be the sum of the best score for each row of the matrix.

The final score is computed as follows:

$$score(e_w) = \frac{\sum_{p_{e_d}}(w_{p_{e_d}} * sum_{p_{e_d}})}{\sum_{p_{e_d}}(w_{p_{e_d}} * \left|V(p_{e_d})\right|)}$$

where $w_{p_{e_d}}$ is the weight assigned to a certain property mapping. If the score is above 0.5[7], the entity is accepted as a web alias for the desktop entity.

5 Evaluation

To evaluate our system, we wanted to measure the accuracy of the matches, in a real-world set-up, with real data. For this purpose we created two entity corpora, one with desktop data and one with web data. To assess the results returned by our system we created first a baseline from relevance judgements made by human experts, on these corpora. Then, we ran our entity matching algorithm and we computed precision, NDCG and MAP to measure its performance.

5.1 Data collection

We created two corpora for the evaluation, one containing desktop entities, and one containing possible matching entities from the Web of Data.

Desktop data entity corpus. The desktop data used in the evaluation was collected from a real, in-use Nepomuk-KDE Semantic Desktop. It was generated by Nepomuk applications, and extracted from the desktop repository.

We restricted the entities selected to three types: (i) people — of type `nco:PersonContact`, (ii) publications — of type `nfo:PaginatedTextDocument`, and (iii) music albums — `nmo:MusicAlbum`. From each type we collected fifty different resources, resulting in a corpus of 150 seed desktop entities, and other entities related to them. Examples of auxiliary entities are the authors of publications, which may or may not be already in the corpus as contacts, the tracks of the albums and the artists. In total the desktop data corpus has 11.917 triples.

We used information from our desktops, therefore the people are colleagues or other researchers we collaborate with; the publications are related to our research interests, and generally related to semantics and information extraction. The music albums data was gathered from several colleagues, for variety of genres.

The contact data is extracted by Nepomuk from the default KDE address book, and we made no changes to it. The correct way to use the `nco:PersonContact` resources extracted automatically, is to link each of them to a corresponding `pimo:Person` representing the person that has the contact information. However, the current tools do not make the distinction, therefore we also used the "raw" `nco:PersonContact` resources, for simplicity. The algorithm makes no distinction between types, so it would yield identical results if we would have used the "proper" `pimo:Person`.

The information related to music albums is extracted automatically by Nepomuk from the ID3 tags of music files.

For publications we used existing tools to perform shallow metadata extraction from files to obtain the title and the authors of the publications, when the metadata of the documents was not set.

[7] We found that the threshold 0.5 was providing better results in our experiment.

Web of Data entity corpus. We used the Sindice query module of our system to generate the second corpus, containing Web of Data entities. For each desktop entity we retrieved the first twenty results returned by Sindice, thus making a total of 3.000 URIs. The queries used in Sindice were constructed as presented in Section 4.1, a combination based on the properties of each desktop entity. For each URI we obtained all the triples extracted by Sindice — explicit and implicit. In total this corpus has 1.530.686 triples.

In this dataset we did not explicitly retrieve Sindice data for the auxiliary entities related to the result URIs. We assumed this data will be available when/if required — in the relevance judgements by experts, and in the matching process by the algorithm.

5.2 Relevance Judgements from Experts

We collected the relevance judgements from experts through an online experiment, in which we asked participants to decide if pairs of desktop and web URIs identify the same real-world object or person. We evaluated in this way all 3.000 pairs from the two corpora. Each pair was judged by three different experts. Eighteen people participated in the experiment, all researchers in the area of Semantic Web.

Fig. 3. The web interface of the experiment for collecting relevance judgements.

To simplify the task, we presented the two entities side by side, with all the information which was available about them in the corpora (see Figure 3). The desktop entity is shown on the left, and the web entity on the right. On the web side we included hyperlinks to the related entities, for further exploration

in the case when the information available was not enough to make the decision. For convenience, on the web side, we have separated and brought to the top the triples which partially matched any of the values from the desktop side.

There were only two decisions possible: *Yes* or *No*, with a *Skip* option, in case of uncertainty. Once a pair was judged or skipped, another one was shown to the participant. The pairs were randomly chosen from the remaining set. To make the experiment feel like a game, we kept count of the number of pairs judged by each participant, and displayed it on the page. We found that even such a small addition generated ad-hoc competition and made the dull task more interesting.

	κ	σ	Avg
All	0.638	0.214	92.252
People	0.661	0.257	88.2
Publications	0.786	0.127	98.067
Albums	0.442	0.233	90.523

Table 1. Inter-annotator agreement measures

The results of the experiment show an average agreement and its standard deviation, computed with Fleiss's κ, of 0.638 ± 0.214, over all three types of entities, suggesting substantial agreement between annotators. Table 1 shows the Fleiss's κ and its standard deviation σ per type, as well as the average pairwise percent agreement. We observed that for music albums, there was only moderate agreement between annotators, visibly lower than the average, while for publications it is visibly higher. We believe the difference is caused by the fact that the data about publications is generated and curated by experts in the field — even more so, as the publications were largely from the domain of Semantic Web —, while the music data comes from much more heterogeneous sources.

5.3 Evaluation Results of the Matching Algorithm

To evaluate the performance of the algorithm, we evaluate each of the matching modules separately and using a combination of them, against a baseline which is the matching framework without any matching modules activated. In the following, the String Matching module is denoted by SM, the Weigthed Properties by WP and the Multi-Valued Properties by MVP.

We used the *trec_eval* tool[8] to compute standard information retrieval measures. The precision at k ($P@k$) with k=1,2,3,4,5, mean average precision (MAP) and normalized discounted cumulative gain (NDCG) are reported in Table 2 for music albums, Table 3 for people and Table 4 for publications. We report also

[8] `http://trec.nist.gov/trec_eval/`

	MAP	NDCG	P@1	P@2	P@3	P@4	P@5
SM WP MVP	0.2464	0.5117	1	0.625	0.4167	0.3125	0.25
SM WP	0.2464	0.5117	1	0.625	0.4167	0.3125	0.25
SM MVP	0.2464	0.5117	1	0.625	0.4167	0.3125	0.25
WP MVP	0	0	0	0	0	0	0
SM	0.2464	0.5117	1	0.625	0.4167	0.3125	0.25
WP	0	0	0	0	0	0	0
MVP	0	0	0	0	0	0	0
Baseline	0	0	0	0	0	0	0

Table 2. Evaluation results for albums, when varying configuration parameters.

the interpolated precision at recall cut-off points when all matching modules are activated. The goal for the system is high precision, i.e., achieving a maximum at P@1. Recall is not a target, as it is generally impossible to determine the entire set of correct results available in the Web of Data.

In Table 2, we can observe that only the SM module is enhancing the results compared to the baseline. The baseline and the other two modules do not help the system at matching certain candidates. Also, in term of MAP and NDCG, the system achieves the lowest performance on the albums corpus. This can be explained by the fact that the album entities are mostly matching entities representing e-commerce products, which are not defined as a type of interests, and therefore rejected by the system. Whether or not such candidates should have been kept by the system is open to discussion and left for a future work.

In Table 3, we can observe that the baseline, the WP and the MVP modules are each one able to match good candidates with high precision at P@1, with WP providing slightly better MAP and NDCG. However, the system does not

	MAP	NDCG	P@1	P@2	P@3	P@4	P@5
SM WP MVP	0.4212	0.6354	0.9302	0.8953	0.7597	0.6337	0.5442
SM WP	0.4174	0.6321	0.9286	0.8929	0.746	0.6131	0.5286
SM MVP	0.4212	0.6354	0.9302	0.8953	0.7597	0.6337	0.5442
WP MVP	0.2916	0.5338	1	0.8243	0.6036	0.473	0.3838
SM	0.4212	0.6354	0.9302	0.8953	0.7597	0.6337	0.5442
WP	0.2916	0.5338	1	0.8243	0.6036	0.473	0.3838
MVP	0.2877	0.53	1	0.8243	0.6036	0.4662	0.3784
Baseline	0.2877	0.53	1	0.8243	0.6036	0.4662	0.3784

Table 3. Evaluation results for people, when varying configuration parameters.

	MAP	NDCG	P@1	P@2	P@3	P@4	P@5
SM WP MVP	0.7773	0.8651	1	0.625	0.4167	0.3125	0.25
SM WP	0.8032	0.8609	0.9062	0.5781	0.3958	0.3047	0.2438
SM MVP	0.7175	0.7986	0.9231	0.5769	0.3846	0.2885	0.2308
WP MVP	1	1	1	0.5	0.3333	0.25	0.2
SM	0.7265	0.7883	0.8235	0.5294	0.3627	0.2868	0.2294
WP	0.6893	0.7347	1	0.55	0.3667	0.275	0.22
MVP	0	0	0	0	0	0	0
Baseline	0.7175	0.7588	1	0.5455	0.3636	0.2727	0.2182

Table 4. Evaluation results for publications, when varying configuration parameters.

get significant advantage by combining them. The SM module alone provides slightly lower precision at P@1 but significantly better MAP and NDCG. By combining the three modules, the system does not get significant advantage and it seems that the SM module prevails.

In Table 4, the baseline provides good results from the start. The system is not able to return any candidates when the MVP is activated by itself. However, when WP and MVP are combined, the system achieves much better results (in term of MAP and NDCG) than the baseline or than the WP module alone. When the system combines the SM module with the two previous ones, the system achieves a lower MAP and NDCG but an improved precision with a larger cut-off rank. While on the two previous types of entities, the SM module seemed to be the most important matching feature, this corpus shows that the WP and MVP are important matching features in certain cases.

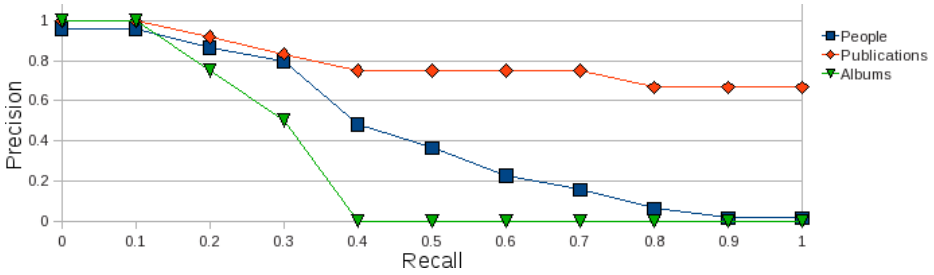

Fig. 4. Interpolated precision at recall cut-off points.

Overall, the results are satisfying for our use cases where high precision prevails over recall. However, given the results shown in Figure 4, we can see that the system could be configured to return more than one entity in order to achieve a better recall while keeping a good precision. It can prove useful to implement

a semi-automatic system which presents the top n candidates to the user for manual selection.

5.4 Performance

To determine the performance, we measure the time spent on each step of the algorithm. To be noted that these results come from a prototypical implementation, still to be subject to technical optimisations. Table 5 shows the average times overall, and for each resource type separately, when all three parameters are active — SM, WP, MVP. We find only small variations in the measurements when the parameter values are changed. We do not consider the time spent on retrieving data from Sindice, as it depends on external factors, like network speed and server availability.

	Overall	People	Publications	Albums
Pair total	375.04	52.19	977.87	53.18
Types check	0.23	0.26	0.21	0.23
Per property check	6.66	0.92	13.2	22.06
All properties	2026.22	7.17	5478.87	1963.88

Table 5. Time performance (milliseconds).

The checking of types is the only value that in average does not depend on the type of resource, as it must be performed for all pairs. The time spent in average per property check is low, but it varies by type, and by the complexity of the properties (e.g. takes longer if several resources in the graph must be traversed, for long property paths like the name of the artist of an album). The "All properties" row shows the average time required for checking all the properties of an entity, and the computation of the final score[9]. These values depend on the type of resources as well, and on the complexity of the resource graph. We found that longer times correspond to very big graphs for online entities, e.g., the graph for `http://webconf.rkbexplorer.com/models/iswc-aswc-2007-complete.rdf`, which must be loaded for checking even if in most cases are not found to represent valid candidates.

6 Discussion and Future Work

The scope of the system presented here is limited to finding Web of Data aliases for desktop resources. We leave the use of the aliases found to future work,

[9] The "All properties" row has values higher that the "Pair total" row because the average time is computed only for those pairs who passed the type check, thus less in number, but with longer computation times.

but the use cases include personalized desktop services like those described in Section 1 and enhancement of desktop information from online sources. We plan to develop a semi-automatic service that retrieves information from the web aliases and updates the local resources, while saving provenance information for the imported data and allowing synchronization when the web data changes.

There exist already web applications that provide similar services via specific APIs (e.g., last.fm). However this is not the goal of this work. Instead, we wish to leverage information across all public information sources accessible on the Web of Data. In addition, such third-party APIs are seen as an additional information sources on the web, and are supported by our system.

Within the system, we make use of existing semantic technologies, one of which are semantic search engines such as Sindice. In the process of determining the aliases we focus on selecting the most appropriate URI from the list of candidates returned by the search engine. In this case, the issues of data sources to trust is left to the search engine, that usually employs advanced techniques [5] for measuring the popularity of a data source. This is however not a requirement we impose on the users, who can choose to query other trusted data sources suitable for their use case.

The system we presented is automatic from the user's point of view, as there are no interactions required for it to work. Once set up it will find and save aliases to desktop resources. Power users can however tweak the settings to fit their specific needs by enabling/disabling modules, changing threshold values or managing mappings. Although the mappings were written manually, they are part of the system and do not need to be modified by end users. We envision for the future, a way of allowing power users to publish their own mappings and let other users install new mappings in a way similar to installing add-ons to web browsers.

7 Conclusion

In this paper, we have presented a framework to automatically link entities from the semantic desktop to the Web of Data. The framework uses existing technologies such as semantic search engines or SPARQL endpoints for retrieving a set of candidates. Each candidate is then evaluated more precisely based on a collection of matching components using string matching, heuristics and rule-based mechanisms. We evaluate qualitatively the system using real-world data retrieved from a Nepomuk Semantic Desktop and the Sindice search engine. The evaluation is based on relevance judgements from a group of experts. We show that the system in its current form provides satisfactory results in term of precision for automatic linking of entities.

Acknowledgments

The work presented in this paper was supported by the Líon-2 project funded by Science Foundation Ireland under Grant No. SFI/08/CE/I1380, and by the European

projects Digital.me (No. 257787) and LOD2 (No. 257943) under the Seventh Framework Program (FP7/2007- 2013).

References

1. Benjelloun, O., Garcia-Molina, H., Jonas, J., Su, Q., Widom, J.: Swoosh: A generic approach to entity resolution. Tech. rep., Stanford University (2006)
2. Bernardi, A., Decker, S., van Elst, L., Grimnes, G., Groza, T., Jazayeri, S.H.M., Mesnage, C., Moeller, K., Reif, G., Sintek, M.: The Social Semantic Desktop: A New Paradigm Towards Deploying the Semantic Web on the Desktop. IGI Global (2008)
3. Bizer, C., Volz, J., Kobilarov, G., Gaedke, M.: Silk - a link discovery framework for the web of data. In: Proceedings of the 18th International World Wide Web Conference (April 2009)
4. Bouquet, P., Stoermer, H., Giacomuzzi, D.: OKKAM: Enabling a web of entities. In: Proceedings of the WWW2007 Workshop I3: Identity, Identifiers, Identification, Entity-Centric Approaches to Information and Knowledge Management on the Web (May 2007)
5. Delbru, R., Toupikov, N., Catasta, M., Tummarello, G., Decker, S.: Hierarchical Link Analysis for Ranking Web Data. In: Proceedings of the 7th Extended Semantic Web Conference (ESWC 2010). pp. 240–256. Springer (2010)
6. Dong, X., Halevy, A.Y., Madhavan, J.: Reference reconciliation in complex information spaces. In: Özcan, F. (ed.) SIGMOD Conference. pp. 85–96. ACM (2005)
7. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: A survey. IEEE Transactions on Knowledge and Data Engineering 19(1), 1–16 (2007)
8. Fellegi, I.P., Sunter, A.B.: A theory for record linkage. Journal of the American Statistical Association 64(328), 1183–1210 (1969)
9. Hogan, A., Harth, A., Decker, S.: Performing object consolidation on the semantic web data graph. In: Proceedings of the WWW2007 Workshop I3: Identity, Identifiers, Identification, Entity-Centric Approaches to Information and Knowledge Management on the Web (May 2007)
10. Jaffri, A., Glaser, H., Millard, I.: URI identity management for semantic web data integration and linkage. In: 3rd International Workshop On Scalable Semantic Web Knowledge Base Systems. Springer (November 2007), http://eprints.ecs.soton.ac.uk/14361/
11. Raimond, Y., Sutton, C., Sandler, M.: Automatic interlinking of music datasets on the semantic web. In: Proceedings of the Linked Data on the Web workshop, LDOW2008 (2008)
12. Saïs, F., Pernelle, N., Rousset, M.C.: L2r: a logical method for reference reconciliation. In: AAAI'07: Proceedings of the 22nd national conference on Artificial intelligence. pp. 329–334. AAAI Press (2007)
13. Sauermann, L., Elst, L.V., Möller, K.: Personal Information Model (PIMO). Deliverable 1.1 (February 2009), http://www.semanticdesktop.org/ontologies/2007/11/01/pimo/v1.1/pimo_v1.1.pdf
14. Tummarello, G., Delbru, R., Oren, E.: Sindice.com: Weaving the open linked data. In: Proceedings of the 6th International Semantic Web Conference. pp. 552–565 (2007)