Department of Information Engineering and Computer Science DISI

UNIVERSITY OF TRENTO - Italy

DISI - Via Sommarive 14 - 38123 Povo - Trento (Italy)
http://www.disi.unitn.it

DISTRIBUTED NAME-BASED ENTITY SEARCH

Fausto Giunchiglia and Alethia Hume

September 2012

Technical Report # DISI-12-033

Accepted at the workshop on Discovering Meaning on the Go in Large Heterogeneous Data 2012 (LHD-12), at the 11th International Semantic Web Conference (ISWC) 2012.

Distributed Name-based Entity Search

Fausto Giunchiglia and Alethia Hume

Department of Information Engineering and Computer Science
University of Trento, Italy
{fausto,hume}@disi.unitn.it
http://www.disi.unitn.it

Abstract. Internet can be seen as a network of peers that store digital representations of entities from the real world (e.g., person, locations, events). Different peers locally represent different "versions" (i.e., different points of view) of the same real world entity. In these different versions, entities are normally identified by multiple (possibly different) names. We propose a *distributed entity search based on names* that aims to (i) find all the different versions of an entity starting from any name used somewhere in the network to identify such entity; and (ii) allow peers to have full control over the privacy of their local representations. We evaluate our approach by setting up a network of 150 peers on PlanetLab. The results show that the performance of our algorithms is stable with the network growth, which is promising in terms of scalability.

Keywords: Entity Search, Named Entities, P2P Networks, DHT

1 Introduction

We see Internet as a network of peers (a P2P network) where peers digitally represent *entities* that exist in the real world. They can be of different types (e.g., person, location, event and others), they have a name, and are described by attributes (e.g., latitude-longitude, size, birth date), which are different for different entity types [2]. Peers represent their own "versions" of entities, i.e., different points of view, showing possibly different aspects of them. These different versions show that the information about entities is inherently distributed.

Entities are normally referred by their names (e.g., Fausto Giunchiglia, Trento, Italy, University of Trento), which play a different role from the other attributes because they are identifiers rather than descriptions [9]. The values of other types of attributes have a meaning that can be understood, e.g., by mapping them to concepts from a knowledge base, like WordNet[1]. Names, on the other hand, are strings that behave very similarly to keywords. Variations and errors on names makes peers use multiple (possibly different) names in their local representations to identify the same real world entity (e.g., Fausto Giunchiglia vs. F. Giunchiglia and Italy vs. Italia). These local names, make search of entities by name difficult.

[1] http://wordnet.princeton.edu/

The reason is that for a given query, only the local representations that uses the exact same name given in the query will be found.

In this paper, we propose a search algorithm that offers the following features:

- First, it takes into account the fact that different names can be used in the local representations of peers to identify the same real world entity. As a result, any name that is used in some local representation to identify an entity can be used to find all the different versions of that entity that are stored in the network of peers. Consider the example of the names *Trento* and *Trient* being used on different entity representations (i.e., the name of the same city but written in different languages). Then, when the peer issues a query that contains one of the names, let us say *Trient*, our approach is able to find both representations.
- Second, it incorporates into the search the notion of a real world entity described by different local representations. Our approach allows peers to locally represent their own versions of the entities of their interest and to have full control on their data. In contrast to current applications (e.g., Facebook, Google Plus, Linkedln, among others), which allow finding information only if it has been uploaded to the central social network server, our approach can find local representations stored on the peers. Only the names of the entity and a link to the local representation are published in the network.

The paper is structured as follows. Section 2 provide the formalization of the notion of entity. In Section 3, we explain the distributed search based on names. Then, Section 4 presents the implementation and evaluation details. In Section 5, we discuss the related work and the conclusions are presented in Section 6.

2 A World of Entities

A Real World Entity (WE) is modeled as a class of local representations that we call Digital Entities (DEs). DEs are defined by different peers and provide a local description of the same entity from the real world. We use a URI (Uniform Resource Identifier) to identify each WE and a URL (Uniform Resource Locator) to identify each DE in the peer. Each URL can be used (by dereferencing) to obtain the full local description (i.e., based on attributes). On the other hand, the peers locally use a non-empty set $\{N\}$ of names to identify a DE. Formally,

$$WE = \langle URI, \{URL\}\rangle \qquad \text{and} \qquad DE = \langle URL, \{N\}\rangle,$$

where $\{URL\}$ is a non-empty set of identifiers of different DEs that describe WE. An example of formalization of entities is shown in Figure 1.

Note that the names are human readable identifiers, which are given to a WE by the peers that locally used them (i.e., the names) in the DEs. Names are labels composed by a combination of words, numbers and symbols and are subject to different types of variations:

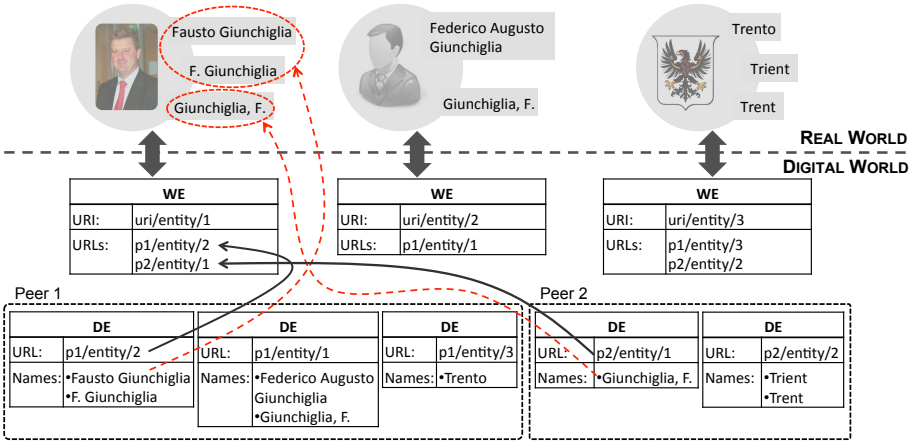

Fig. 1. Examples of entity formalizations

- **Order.** The words of a name can be written in different order (e.g., *Fausto Giunchiglia* and *Giunchiglia, Fausto*).
- **Abbreviations.** Multiple abbreviations can exist for the same full name (e.g., *Federico Augusto Giunchiglia* can be abbreviated as *F. A. Giunchiglia*, *Federico A. Giunchiglia* and others). On the other hand, the abbreviation of a name can be a valid reference to many different full names (e.g., *F. Giunchiglia* is valid for *Fausto Giunchiglia* but also for *Federico Giunchiglia*).
- **Nicknames.** Arbitrary nicknames are sometimes used by peers to refer to a *DE* (e.g., *Fede* can be used as a nickname for *Federico* and also for *Federica*).
- **Translations.** Names sometimes change (are written differently) in different languages (e.g., *Trento* in Italian, *Trient* in German or *Trent* in English).
- **Misspellings.** Names can be misspelled, either in the definition of a *DE* or during the specification of a search query (e.g., *Fasuto* instead of *Fausto*).

The name variations together with the DE definition presented above, show that the relation between names and *DEs* is of the type *many-to-many*. In turn, this leads to a name-matching problem when we intend to search an entity based on its names, see [9] for a more detailed discussion of the issues related to the name matching problem. On the other hand, the relation between *DEs* and *WEs* is of the type *one-to-many*. This is a consequence of the fact that one *DE* can represent only one entity from the real world (i.e., one *WE*) and a *WE* can be represented by different *DEs* stored on different peers. Moreover, Figure 1 shows that names are not unique, i.e., different *WEs* can be called by the same name (e.g., when we have homonyms). We can also see that different *DEs* can give different names to the same *WE*. As a result, names and *WEs* can also be associated with a relation of the type *many-to-many*.

3 Distributed Name Search

Our goal is to address the problem of searching entities based on their names. Formally, we define a query as $Q = \{N^Q\}$, where $\{N^Q\}$ is the non-empty set of names used to identify one target WE. Many candidates for the correct answer can be found as a consequence of the many-to-many relation between names and WEs. Let URL^{DE} and $\{N^{DE}\}$, be the identifier and the set of names of a digital entity DE. The general problem of the *Distributed Name-based Entity Search* can be seen as retrieving WEs that are described in the network by at least one DE, such that, the intersection between $\{N^{DE}\}$ and $\{N^Q\}$ is not empty. Formally, the Query Answer (QA) can be defined as:

$$QA = \{WE \mid \exists\, URL' \in WE \; s.t., URL' = URL^{DE} \;\wedge\; \{N^{DE}\} \cap \{N^Q\} \neq \emptyset\}$$

This definition considers a partial matching between $\{N^{DE}\}$ and $\{N^Q\}$ in order to allow finding a WE from any of the names given to it on different DEs.

The notions of WE and DE, introduced in section 2, allow the separation of the problem of taking into account multiple names for a WE, and the problem of finding the DEs that represent different versions of a WE. In turn, this separation can be used to split the QA in two sub-problems:

1. Searching WEs that match with the names in $\{N^Q\}$
2. Searching DEs that correspond to a given WE

In order to search WEs based on names, we define a $Name\_Index$, which stores mappings between names and WEs (i.e., the $URIs$ of the WEs) that are identified by such names:

$$\{N\} \Rightarrow \{URI\}$$

This index encodes the *many-to-many* relation between names and WEs. The dynamics of the index is given by the publication and the deletion of DEs (from peers) in the network. When the peer publishes a DE, the mapping between each name N_i^{DE} in $\{N^{DE}\}$ and the WE that is associated to the DE is added to the index. We assume that the peer locally caches the identifier (i.e., the URI) of the WE that is represented by its DE[2]. On the other hand, the deletion of a DE from the network does not directly affect the $Name\_Index$. A mapping between N_i^{DE} and WE can be removed from the index only when there are no DEs in the network that represent such WE. Periodic checks are performed over the $Name\_Index$ in order to detect this situation and remove such mappings. In the first step of the search, this index allows taking advantage of the many local names specified in different DEs to find the WEs that are related to the names given in $\{N^Q\}$. For example, let us suppose that a peer issues a query $Q = \{Giunchiglia, F.\}$. The $Name\_Index$ is used to find the relevant WEs, i.e., $uri/entity/1$ and $uri/entity/2$.

In order to search DEs that represent the same WE, we define a WE_Index that store information about the different versions of a WE. The index store

[2] Note that the initial identification of the WE described by a DE is a problem of identity management and is out of the scope of this work.

mappings between each WE (i.e., URI of the WE) and the DEs (i.e., $URLs$ of the DEs) that represent it:

$$URI \Rightarrow \{URL\}$$

This index is affected by the publication and deletion of DEs in a straightforward manner. The mapping of a WE and a DE that represent it, is added to the WE_Index when a peer publishes the DE. The same mapping is removed from the index when the peer deletes the DE. In the second step of the search, the WE_Index is used to complete the query answer with the different DEs that are stored on peers and describe the WEs found in the first step. Consider the previous example and suppose that its output is used as input here, i.e., $uri/entity/1$ and $uri/entity/2$. Next, the Uri_Index is used to get the set $\{\langle uri/entity/1, \{p1/entity/2, p2/entity/1\}\rangle, \langle uri/entity/2, \{p1/entity/1\}\rangle\}$ of WEs that complete the QA for the query $Q = \{Giunchiglia, F.\}$.

4 Implementation and Evaluation

The indexes are stored in the P2P network by extending the basic functions of Distributed Hash Tables (DHTs)[3]. We use a DHT library called TomP2P[4], which allow us to (i) store multiple values mapped to the same key; and (ii) execute the operations over different index domains. The different index domains can be seen as having one DHT for the $Name\_Index$ and other for the WE_Index.

In order to evaluate the proposed approach, we conducted a set of experiments where the scalability of our search algorithm was analyzed. We are interested on measuring how much the performance (considered in terms of the query processing time) is affected by the network growth. Networks of different sizes were set up for the evaluation using *PlanetLab*[5], a network of computers (i.e., nodes) available as a testbed for research. For the generation of data-sets, we used data from the proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)[6], which are available online. We extracted the titles of publications, names of authors and names of locations related to the conference.

Three data-sets were generated to populate networks of 50, 100 and 150 peers. Each data-set was produced by generating triples of $\langle Name, URI, URL\rangle$. Names and $URIs$ were replicated in order to simulate different WEs having the same name and different peers storing DEs that describe the same WE. Let us call p_n to the popularity of a name n (i.e., number of WEs that are called by n) and p_{we} to the popularity of a WE (i.e., number of DEs that represent a WE). First, for each name n, we generated p_n triples with the same name (different URI and URL). Second, for each URI, we generated p_{we} triples with the same name and URI but with different $URLs$. The popularities p_n and p_{we} follow a Zipf distribution, which means that there is a long tail of unpopular names

[3] http://en.wikipedia.org/wiki/Distributed_hash_table
[4] http://www.tomp2p.net/
[5] https://www.planet-lab.eu/
[6] http://ijcai.org/

and WEs. The distribution of both popularities are independent, which means that a popular WE do not necessarily has a popular name and vice versa. We assume that the local entity base of each peer contains, in average, 2000 DEs. We had overall around 100000, 200000 and 300000 DEs. The query set for each peer was generated by randomly selecting a set of 1400 names from the initial set of entity names.

We registered the time that the system takes to respond to each query executed by the peers. Then, we computed the average query time for the network. The average query processing times for the different networks are shown in Table 1. We can see that, although the average times are high (in particular if we compare them to what is expected from an information retrieval system), they are maintained stable in the different sizes networks.

Table 1. Average query time

Network Size	50 peers	100 peers	150 peers
Avg. Query Time (in seconds)	2.77	2.75	2.61

In Figure 2, we show the distribution of the query time in the different networks. We can see that more than 55% of the queries are actually answered in less than a second, while in almost 70% of the cases the response arrives in less than 2 seconds (which is less than the average time). Moreover, only 9% of queries take more than 5 seconds to be answered. Another interesting remark is that also the query time distribution is stable with regard to the network growth.

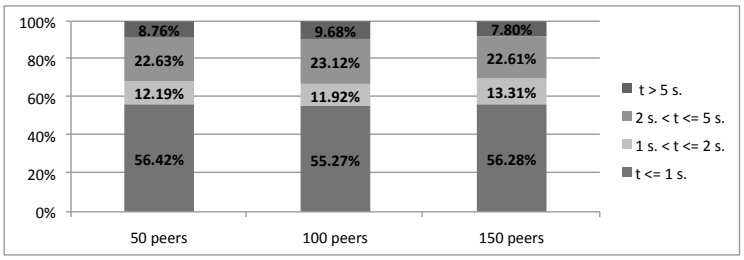

Fig. 2. Query time of different networks

In the current implementation, the DHT lookups to answer a query are executed sequentially. Moreover, all the lookups have to end before returning any result and a perfect matching between the names in the query and the names in the index is assumed. On the other hand, we need to consider that slow DHT lookups can be a consequence of some slow peers that participate in the network. We foresee that parallelization, result catching and existing techniques that avoid routing through slow peers (see for example [16]) can be implemented in order to reduce the absolute query time. Because of this, we believe that the stability in the performance of the approach is a promising result.

5 Related Work

The search approach presented in this paper combines the areas of entity search and P2P systems. To the best of our knowledge there are no approaches that integrates these areas, i.e., that performs search of entities over a p2p network. Existing entity aware approaches concentrate the attention on the definition of models and structures for the representation of entities. In [2, 3], the Semantic Web is seen as a global space into which the semantic knowledge from different sources is integrated. Few approaches that return entities as search result can be found in the literature [4, 10], but in both approaches the search is centralized. In contrast to these approaches, our approach performs a distributed search in a P2P network and allows users to maintain their data locally.

On the other hand, we have P2P approaches, which perform distributed search but are not aware of entities [17, 14]. They are mainly classified as unstructured and structured approaches. The first unstructured networks (e.g., Gnutella[7]) have scalability problems due to the number of messages generated and do not guarantee that all answers will be found. Other approaches use clustering techniques [1, 5, 18, 6, 12], their goal is to find the best group to answer a query and then send the query to the peers in that group. Our approach can find all available answers and has proven to be promising in terms of scalability.

Structured approaches aim to guarantee the location of the content shared on the network (e.g., CAN[8], Chord[9], Pastry[10] and Tapestry[11]). They store pairs of $\langle key, value \rangle$ in a Distributed Hash Table (DHT) and then retrieve the value associated with a given key. Other approaches perform multi-keyword search using DHTs but they can be very expensive in terms of required storage and generated traffic (e.g., see [13]). Hierarchical structures combine clustering techniques with the structure of DHTs [7, 11, 15, 8]. In general, P2P approaches provide the techniques needed in order to build our solution. The novelty of our approach is in the domain of application of such techniques.

6 Conclusions

We presented an approach that can find different versions of an entity, which are stored on different peers, from any name that is used in the network to identify such entity. Moreover, our algorithm allow peers to have full control over the privacy of their data. We evaluated the search on networks of 50, 100, and 150 peers running on PlanetLab. The approach shows evidence of being scalable because the performance is stable with the different network sizes.

The current algorithm considers the name variations used by the peers in the network. In the future we want to extend this work by studying the name

[7] `http://en.wikipedia.org/wiki/Gnutella`
[8] `http://en.wikipedia.org/wiki/Content_addressable_network`
[9] `http://en.wikipedia.org/wiki/Chord_(peer-to-peer)`
[10] `http://en.wikipedia.org/wiki/Pastry_(DHT)`
[11] `http://en.wikipedia.org/wiki/Tapestry_(DHT)`

matching problem in more detail in order to support the disambiguation of queried names that do not match with any of the names used in the network. This should also help to improve the ranking of search results. Additionally, our next steps include the parallelization of our algorithm, the study of techniques that help improving the performance of the search and further tests with more realistic data-sets.

References

1. M. Bawa, G. Manku, and P. Raghavan. Sets: Search enhanced by topic segmentation. In *Proceedings of ACM SIGIR Conference*, pages 306–313, 2003.
2. B. Bazzanella, J. A. Chaudhry, Themis Palpanas, and H. Stoermer. Towards a General Entity Representation Model. *5th Workshop on SWAP*, 2008.
3. P. Bouquet, H. Stoermer, C. Niederee, and A. Maña. Entity name system: The back-bone of an open and scalable web of data. In *Proceedings of the 2nd IEEE ICSC*, pages 554–561, Washington, DC, USA, 2008. IEEE Computer Society.
4. T. Cheng and K. C.-C. Chang. Entity search engine: Towards agile best-effort information integration over the web. In *CIDR 2007*, pages 108–113, 2007.
5. E. Cohen, H. Kaplan, and A. Fiat. Associative search in peer to peer networks: Harnessing latent semantics. In *Proceedings of IEEE INFOCOM*, 2003.
6. A. Crespo and H. Garcia-Molina. Semantic overlay networks for p2p systems. Technical report, Stanford University, 2002.
7. P. Ganesan, K. Gummadi, and H. Garcia-Molina. Canon in g major: designing dhts with hierarchical structure. In *ICDCS'04*, pages 263 – 272, 2004.
8. L. Garcés-Erice, E. W. Biersack, P. Felber, K. W. Ross, and G. Urvoy-Keller. Hierarchical peer-to-peer systems. In *Euro-Par*, pages 1230–1239, 2003.
9. G. Holloway and M. Dunkerley. *The Math, Myth and Magic of Name Search and Matching*. Search Software America, 5th edition, 2004.
10. G. Hu, J. Liu, H. Li, Y. Cao, J.-Y. Nie, and J. Gao. A supervised learning approach to entity search. In *AIRS'06*, volume 4182 of *LNCS*, pages 54–66. 2006.
11. D. Janakiram, F. Giunchiglia, H. Haridas, and U. Kharkevich. Two-layered architecture for peer-to-peer concept search. In *4th Int. Sem Search Workshop*, 2011.
12. S. Joseph. Neurogrid: Semantically routing queries in peer-to-peer networks. In *Proc. Intl. Workshop on Peer-to-Peer Computing*, pages 202–214, 2002.
13. J. Li, B. Thau, L. Joseph, M. Hellerstein, and M. F. Kaashoek. On the feasibility of peer-to-peer web indexing and search. In *IPTPS'03*, 2003.
14. E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys and Tutorials*, 7:72–93, 2005.
15. O. Papapetrou, W. Siberski, and W. Nejdl. Pcir: Combining dhts and peer clusters for efficient full-text p2p indexing. *Computer Networks*, 54(12):2019–2040, 2010.
16. S. Rhea, B. Chun, J. Kubiatowicz, and S. Shenker. Fixing the embarrassing slowness of OpenDHT on PlanetLab. *Proc. of the Second USENIX Workshop on Real, Large Distributed Systems*, 0:25–30, 2005.
17. J. Risson and T. Moors. Survey of research towards robust peer-to-peer networks: Search methods. *Computer Networks*, 50:3485–3521, 2006.
18. K. Spripanidkulchai, B. Maggs, and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems. In *Proceedings of IEEE INFOCOM*, volume 3, pages 2166–2176, 2003.