Expert Systems With Applications 44 (2016) 354–366

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Building a relatedness graph from Linked Open Data: A case study in the IT domain

Tommaso Di Noia[a,*], Vito Claudio Ostuni[a], Jessica Rosati[a], Paolo Tomeo[a], Eugenio Di Sciascio[a], Roberto Mirizzi[b], Claudio Bartolini[c]

[a] *Polytechnic University of Bari, via E. Orabona, 4 –Bari 70125, Italy*
[b] *Yahoo! Inc., 701 First Avenue –Sunnyvale,CA94089, USA*
[c] *HP Labs, 1501 Page Mill rd. –Palo Alto,CA94304, USA*

ARTICLE INFO

Keywords:
Relatedness graph
Linked data
DBpedia

ABSTRACT

The availability of encyclopedic Linked Open Data (LOD) paves the way to a new generation of knowledge-intensive applications able to exploit the information encoded in the semantically-enriched datasets freely available on the Web. In such applications, the notion of relatedness between entities plays an important role whenever, given a query, we are looking not only for exact answers but we are also interested in a ranked list of related ones. In this paper we present an approach to build a relatedness graph among resources in the `DBpedia` dataset that refer to the IT domain. Our final aim is to create a useful data structure at the basis of an expert system that, looking for an IT resource, returns a ranked list of related technologies, languages, tools the user might be interested in. The graph we created is a basic building block to allow an expert system to support the user in entity search tasks in the IT domain (e.g. software component search or expert finding) that goes beyond string matching typical of pure keyword-based approaches and is able to exploit the explicit and implicit semantics encoded within LOD datasets. The graph creation relies on different relatedness measures that are combined with each other to compute a ranked list of candidate resources associated to a given query. We validated our tool through experimental evaluation on real data to verify the effectiveness of the proposed approach.

1. Introduction

The emergence of the crowd computing initiative has brought on the Web a new wave of tools enabling collaboration and sharing of ideas and projects, ranging from simple blogs to social networks, as well as software platforms and even mashups. However, when these web-based tools reach the “critical mass” one of the problem that suddenly arises is how to retrieve content of interest from such rich repositories. As a way of example, we may refer to a platform to share software components, where programmers can publish APIs and mashups. When a user uploads a new piece of code, they tag it so that the component will be later retrievable by other users. Components can be retrieved through a keywords-based search or browsing across categories, most popular items or new updates. Most of the current systems usually rely on text matching between a search query and the textual description of a resource or a set of associated tags. A match is found when keywords, or patterns of keywords expressed in the query appear also in the description associated to the resource. However, text-based approaches suffer from the innate problems of ambiguity of natural language (Resnik, 1999). Even if the query and the resource descriptions are somehow structured, the same issues persist. In particular, one of the biggest deficiency in these approaches is their inability to capture the meaning of terms expressed both in the query and in the description, and the semantic relations between such terms. As an example, let us consider some cases where systems based exclusively on text analysis fail, with a particular emphasis on the IT domain:

- Both *SVM* and *Support Vector Machine* refer to the same Machine Learning algorithm. A textual approach by itself is not able to deal with synonymy.
- *Ubuntu* and *Debian* are two Linux distributions, but for a text-based system there is no way to understand how they are related.
- *PHP* and *MySQL* are two different technologies but strongly related with each other. Indeed, *MySQL* is the de facto standard DBMS used when developing *PHP* applications.

[*] Corresponding author. Tel.: +39 3346715671; fax: +39 0805963410.
E-mail addresses: tommaso.dinoia@poliba.it (T. Di Noia), vitoclaudio.ostuni@poliba.it (V.C. Ostuni), jessica.rosati@poliba.it (J. Rosati), paolo.tomeo@poliba.it (P. Tomeo), eugenio.disciascio@poliba.it (E. Di Sciascio), robertom@yahoo-inc.com (R. Mirizzi), claudio.bartolini@hp.com (C. Bartolini).

http://dx.doi.org/10.1016/j.eswa.2015.08.038
0957-4174/© 2015 Elsevier Ltd. All rights reserved.

- *Java* is an *Object-oriented programming language*. The relation *isA* is pretty common when modelling knowledge domains. However, algorithms that are purely based on keywords cannot understand it.

The previous examples stress once more the importance of capturing semantic relations among entities, and of being able to identify (semantically) related resources (together with a relatedness value).

In this paper we show how to tackle such problems by considering entities and their associated semantics instead of simple keywords. In particular, we demonstrate that by leveraging knowledge bases which are freely available in the Web of Data we can compute the *relatedness* between concepts belonging to the IT (Information Technology) domain.

In fact, the notion of relatedness is wider than that of similarity. While the latter refer to specific classes of objects (the class of databases, the class of programming languages, etc.), the former refers to the whole knowledge space (database and programming languages, etc.). By remaining in the IT domain, if we consider *MySQL* and *PostgreSQL* we may say if they are similar or not as they are two DBMSs. On the other hand, if we consider *MySQL* and *PHP* we cannot state anything about their similarity but we can say if they are related with each other.

In this work we present semantic-aware measures to evaluate the relatedness values between IT concepts. Using these measures, we then build a graph where nodes are IT concepts (programming languages, databases, technologies, frameworks, etc.) and edges between nodes indicate they are related with each other. We also associate a numerical label to each edge that represents the relatedness value between two nodes. Having a tool able to measure and evaluate how much two resources are related with each other, is a key factor in the design and development of an expert system able to foster the process of selecting those resources semantically related (to different extents) to the ones that the user is looking for. Indeed, one of the main tasks an expert system must be able to cope with is that of supporting human users in decision-making processes such as "*help me in finding those items better corresponding to my needs*". Within a knowledge space, encoding the notion of relatedness among resources as a graph may, for instance, allow an expert system to: (i) support the users in ranking those resources which relate to the ones they are interested in; (ii) guide the users through an exploratory browsing Marchionini (2006) of the knowledge space by following links whose semantics represents the relatedness degree between the explored nodes.

Our graph is built by leveraging and combining statistical knowledge obtained from the Web and semantic knowledge extracted from the encyclopedic knowledge base DBpedia[1]. We adopt an approach based on machine learning to effectively combine such information. The approach we present here builds on top of Mirizzi, Ragone, Noia, and Sciascio (2010). Nevertheless, there are many differences and improvements. First of all, they rely on a very expensive, from a computational point of view, process for extracting relevant resources from DBpedia. It considers a continuous interaction between the graph exploration and the computation of Web-based conditional probabilities. Moreover, they manually combine different features in a naive way instead of automatically combining them as we propose here.

The novel contributions of this work are listed in the following:

- proposal of a measure for finding the relatedness of concepts in the IT domain, based on statistical, textual and semantic analysis combined via both a Learning to Rank (Liu, 2009) (LTR) and a data fusion (Nuray & Can, 2006) approach;
- construction of a relatedness graph for IT concepts on top of DBpedia;
- experimental evaluation of the approach on real data extracted from job posts.

The remainder of this paper is structured as follows. In the next section we give a brief overview of the Semantic Web technologies we adopt in our approach. In Section 2 we describe the advantages of using semantic knowledge bases and we provide information about DBpedia. The relatedness measure between IT terms and the graph building are detailed in Section 3. In Section 4 we present the results of the evaluation of our approach. Related work is discussed in Section 5. Conclusion and Future work conclude the paper.

2. Linked Data as a knowledge source in the IT domain

If we wanted to build an expert system able to catch relatedness between IT technologies and tools we would have needed a way to capture the meaning behind keywords in order to overcome the issues of text-based approaches. Indeed, in this knowledge-intensive scenario, detailed information about entities plays a fundamental role. During the last few years, the Web has been evolving in the so called Web of Data where the main actors are no more pages identified by a URL but resources/data identified by a URI. In this transformation process the Linked Open Data (LOD) (Bizer, Heath, & Berners-Lee, 2009a) initiative has been a first calls citizen. The *Linking Open Data* community project started in 2007 with the goal of augmenting the current Web with data published according to Semantic Web standards. The idea is to use RDF [2] to publish various open datasets on the Web as a vast decentralized knowledge graph, commonly known as the LOD cloud. As of today, several dozen billion RDF triples are freely available covering diverse knowledge domains and tightly connecting different datasets with each other.

DBpedia. One of the most popular datasets in the LOD compass is DBpedia (Bizer et al. 2009b). It is a community effort to extract structured information from Wikipedia and make it freely accessible as RDF triples. This knowledge base currently describes 4 million resources, out of which 3.22 million are classified in a consistent ontology[3]. Its SPARQL endpoint[4] allows anyone to ask complex queries about such resources. Each element in DBpedia is identified by its own URI in order to avoid ambiguity issues. For example, the programming language *Java* is referred to as the resource identified by the URI dbpedia:Java_(programming_language), whereas the software platform *Java* is identified by the URI dbpedia: Java_(software_platform). The resource dbpedia:Java_ (disambuigation) describes all the possible meanings for the label *Java* thanks to the property dbpedia-owl: wikiPageDisambiguates. Similarly, both the URI dbpedia:Svm_(machine_learning) and the analogous URI dbpedia:Support_vector_machine refer to the same Machine Learning algorithm, hence to the same resource. In DBpedia this relation is captured via the property dbpedia-owl:wikiPageRedirects that connects the former to the latter entity.

Compared to other hierarchies and taxonomies, DBpedia has the benefit that each term/resource is endowed with a rich textual description via the dbpedia-owl:abstract property and at least one textual label via rdfs:label. The value associated to dbpedia-owl:abstract is a string containing the text before the table of contents (at most 500 words) of a Wikipedia page, while the property rdfs:label contains the title of the Wikipedia page. The multilingual nature of Wikipedia is reflected in the values of dbpedia-owl:abstract and rdfs:label. In fact, given a DBpedia URI, we may have a description and a label for each

[1] http://dbpedia.org

[2] See Appendix for a quick overview on RDF and SPARQL.

[3] http://wiki.dbpedia.org/Ontology

[4] http://dbpedia.org/sparql