

Computing Semantic Relatedness using DBpedia

José Paulo Leal¹, Vânia Rodrigues², and Ricardo Queirós³

- 1 CRACS & INESC-Porto LA, Faculty of Sciences, University of Porto
Porto, Portugal
zp@dcc.fc.up.pt
- 2 CRACS & INESC-Porto LA, Faculty of Sciences, University of Porto
Porto, Portugal
c0416098@alunos.dcc.fc.up.pt
- 3 CRACS & INESC-Porto LA & DI-ESEIG/IPP
Porto, Portugal
ricardo.queiros@eu.ipp.pt

Abstract

Extracting the semantic relatedness of terms is an important topic in several areas, including data mining, information retrieval and web recommendation. This paper presents an approach for computing the semantic relatedness of terms using the knowledge base of DBpedia — a community effort to extract structured information from Wikipedia. Several approaches to extract semantic relatedness from Wikipedia using bag-of-words vector models are already available in the literature. The research presented in this paper explores a novel approach using paths on an ontological graph extracted from DBpedia. It is based on an algorithm for finding and weighting a collection of paths connecting concept nodes. This algorithm was implemented on a tool called Shakti that extract relevant ontological data for a given domain from DBpedia using its SPARQL endpoint. To validate the proposed approach Shakti was used to recommend web pages on a Portuguese social site related to alternative music and the results of that experiment are reported in this paper.

1998 ACM Subject Classification H.3.3 Information Search and Retrieval

Keywords and phrases semantic similarity, processing wikipedia data, ontology generation, web recommendation

Digital Object Identifier 10.4230/OASISs.SLATE.2012.133

1 Introduction

Searching effectively on a comprehensive information source as the Web or just on the Wikipedia usually boils down to using the right search terms. Most search engines retrieve documents where the searched terms occur exactly. Although stemming search terms to obtain similar or related terms (e.g. synonyms) is a well known technique for a long time [15], it usually considered irrelevant in general and search engines of reference no longer use it [1].

Nevertheless, there are cases where semantic search, a search where the meaning of terms is taken in consideration, is in fact useful. For instance, to compare the similarity of genes and proteins in bio-informatics, to compare geographic features in geographical informatics, and to relate multi-word terms in computational linguistics.

The motivation for this research in semantic relatedness comes for another application area, recommendation. Most recommenders use statistical approaches, such as collaborative filtering, to make suggestions based on the choices of users with a similar choice pattern. For instance, an on-line library may recommend a book selected by other users that also bought



© José Paulo Leal, Vânia Rodrigues, and Ricardo Queirós;
licensed under Creative Commons License NC-ND

1st Symposium on Languages, Applications and Technologies (SLATE'12).

Editors: Alberto Simões, Ricardo Queirós, Daniela da Cruz; pp. 133–147

OpenAccess Series in Informatics



OASIS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

the books already in the shopping basket. This approach has a cold start issue: what should be recommended to someone that was not yet bought anything? to whom recommend a book that was just published and few people have bought?

An alternative approach is to base recommenders on an ontology of recommend items. An on-line library can take advantage from the structure of an existing book classification, such as the Library of Congress Classification system. However, in many cases such classification does not exist and the cost of creating and maintaining an ontology would be unbearable. This is specially the case if one intends to create an ontology on a unstructured collection of information, such as a folksonomy.

Consider a content-based web recommendation system for a social network, where multimedia content (e.g. photos, videos, songs) is classified by user-provided tags. One could simply recommend content with common tags but this approach would provide only a few recommendations since few content item share the exact same tags. In this case, to increment the number of results, one could search for related tags. For instance, consider that your content is related to music that users tag with names of artists and bands, instruments, music genres, and so forth. To compute the semantic relatedness among tags in such a site one needs a specific ontology adapted to this type of content.

It should be noticed that, although several ontologies on music already exist, in particular the Music Ontology Specification¹, they are not adjusted to this particular use. They have a comprehensive coverage of very broad music genres but lack most of the sub-genres pertinent to an alternative music site. To create and maintain an ontology adjusted to a very specific kind the best approach is to extract it from an existing source. The DBpedia² is a knowledge base that harvests the content of the Wikipedia and thus covers almost all imaginable sources. It is based on an ontology that classifies its subjects and on mapping rules that convert the content of Wikipedia info-boxes and tables into Resource Description Framework (RDF) triplets available from a SPARQL endpoint (SPARQL is a recursive acronym for SPARQL Protocol and RDF Query Language).

In this paper we present Shakti, a tool to extract an ontology for a given domain from DBpedia and use it to compute the semantic relatedness of terms defined as labels of concepts in that ontology. One of the main contribution of this paper is the algorithm used for computing relatedness. Most algorithms for computing semantic relatedness based on ontologies assume that these are taxonomies or at least direct acyclic graphs which is not the case with an ontology extracted from DBpedia. Also, these algorithms usually focus on a notion of distance. Instead the proposed algorithm is based on a notion of proximity. Proximity measures how connected two terms are, rather than how distant they are. A term may be at the same distance to other two terms but have more connections to one than the other. Terms with more connections are in a sense *closer* and thus have an higher proximity.

The rest of this paper is organized as follow. The following section presents related work on semantic relatedness algorithms and on the use of knowledge bases such as DBpedia. Section 3 is the main section as it presents the proposed algorithm and Shakti, a tool implementing it. The following section presents a use of Shakti to populate a proximity table of a recommender service that was used as validation of the proposed approach. The final section summarizes the contributions of this paper and highlights future directions of this research.

¹ <http://musicontology.com/>

² <http://dbpedia.org/About>

2 Related Work

This section summarizes the concepts and technologies that are typically used as basis for the computation of semantic relatedness of terms in the Web.

2.1 Knowledge Representation

Currently, the Web is a set of unstructured documents designed to be read by people, not machines. The semantic web — sponsored by W3C - aims to enrich the existing Web with a layer of machine-interpretable metadata on Web resources so that computer programs can predictably exchange and infer new information. This metadata is usually represented by a general purpose language called Resource Description Framework (RDF). Its specification [2] includes a data model and a XML binding. The data model of RDF is a collection of triples — subject, predicate and object — that can be viewed as a labeled directed multi-graph; a model well suited for knowledge representation. Ontologies formally represent knowledge as a set of concepts within a domain, and the relationships between those concepts. Ontology languages built on top of RDF provide a formal way to encode knowledge about specific domains, including reasoning rules to process that knowledge [4]. In particular, RDF Schema [3] provides a simple ontology language for RDF metadata that can be complemented with the more expressive constructs of OWL [12]. The triplestores can be queried and updated using the SPARQL Protocol and RDF Query Language (SPARQL).

2.2 Knowledge Bases

Knowledge bases are essentially information repositories that can be categorized as machine or human-readable information repositories. A human-readable knowledge base can be coupled with a machine-readable one, through replication or some real-time and automatic interface. In that case, client programs may use reasoning on computer-readable portion of data to provide, for instance, better search on human-readable texts. A great example is the machine-readable DBpedia extraction from human-readable Wikipedia.

Wikipedia articles consist mostly of free text. However, the joint efforts of human volunteers have recently obtained numerous facts from Wikipedia, storing them as machine-harvestable triplestores in Wikipedia infoboxes [17]. The DBpedia project extracts this structured information and combines this information into a huge, cross-domain knowledge base. DBpedia uses RDF as the data model for representing extracted information and for publishing it on the Web. Then, SPARQL can be used as the query language to extract information allowing users to query relationships and properties associated with many different Wikipedia resources.

2.3 Semantic Similarity

Extracting the semantic relatedness of terms is an important topic in several areas, including data mining, information retrieval and web recommendation. Typically there are two ways to compute semantic relatedness on data:

1. by defining a topological similarity using ontologies to define the distance between words (e.g. in a directed acyclic graph the minimal distance between two term nodes);
2. by using statistical means such as a vector space model to correlate words from a text corpus (co-occurrence).

Semantic similarity measures have been developed and applied in several domain ontologies such as in Computational Linguistics (e.g. Wordnet³) or Biomedical Informatics (e.g. Gene Ontology⁴). In order to calculate the topological similarity one can rely either on ontological concepts (edge-based or node-based) or ontological instances (pairwise or groupwise). A well-known node-based metric is the one developed by Resnik [13] which computes the probability of finding the concept (term or word) in a given corpus. It relies on the lowest common subsumer which has the shortest distance from the two concepts compared. This metric is usually applied on WordNet [6] a lexical database that encodes relations between words such as synonymy and hypernymy. A survey [14] between human and machine similarity judgments on a Wordnet taxonomy reveal highest correlation values on other topological metrics such the ones developed by Jiang [9] and Lin [10].

Statistical computation of semantic relatedness relies on algebraic models for representing text documents (and any objects, in general) as vectors of identifiers. Comparing text fragments as bags of words in vector space [1] is the simplest technique, but is restricted to learning from individual word occurrences. The semantic sensitivity is another issue where documents with similar context but different term vocabulary won't be associated, resulting in a "false negative match". Latent Semantic Analysis (LSA) [5] is a statistical technique, which leverage word co-occurrence information from a large unlabelled corpus of text [8].

Currently, Wikipedia has been used for information retrieval related tasks [16, 18, 7, 11]. This is due to the increasing amount of articles available and the associate semantic information (e.g. article and category links). One of these efforts is the Explicit Semantic Analysis(ESA), a novel method that represents the meaning of texts in a high-dimensional space of concepts derived from Wikipedia and the Open Directory Project (ODP). It uses machine learning techniques to represent the meaning of any text as a weighted vector of Wikipedia-based concepts. The relatedness of texts in this space is obtained by comparing the corresponding vectors using conventional metrics (e.g. cosine) [7].

3 Shakti

This section presents an approach to compute the semantic relatedness between terms using ontological information extracted from DBpedia for a given domain. The first subsection outlines the algorithm used for computing semantic relatedness as a measure of proximity between nodes in a graph. The second subsection presents the design of a system to extract the relevant part of DBpedia for a given domain, connect concept nodes on the ontological graph and compute their proximity. The last subsection describes the actual implementation of Shakti.

3.1 Algorithm

Concepts on DBpedia are represented by nodes. Take for instance the music domain used for the case study presented in section in section 4. Singers, bands, music genres, instruments or virtually any concept related to music is represented as a node in DBpedia. These nodes are connected by properties, such as **has genre** connecting singers to genres, and thus form a graph. This graph can retrieved in RDF format using the SPARQL endpoint of DBpedia.

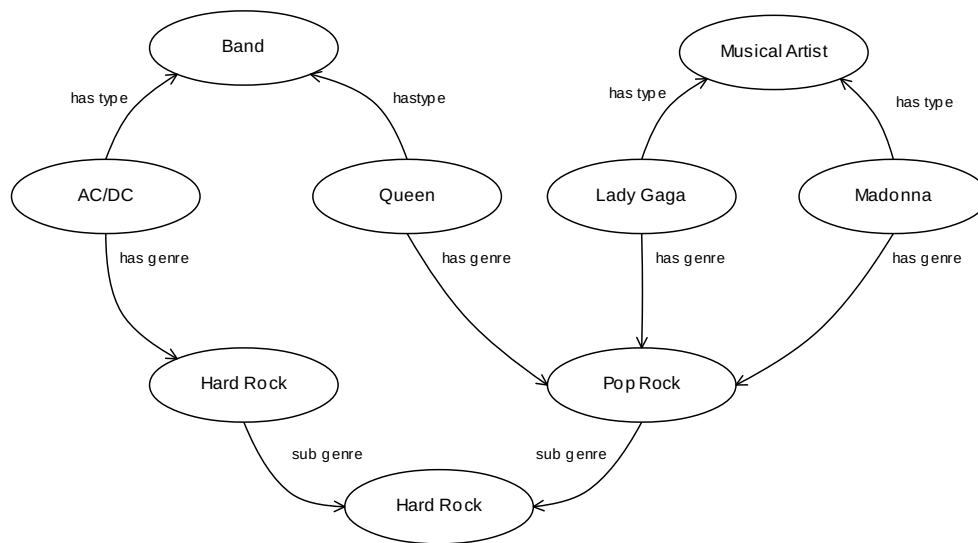
³ <http://wordnet.princeton.edu/>

⁴ <http://www.geneontology.org/>

The core idea in the research presented in this paper is to use the RDF graph to compute the similarity between nodes. Actually, we are interested in the similarity between terms, but each node and arc of this graph has a label — a string representation or stringification — that can be seen as a term.

At first sight relatedness may seem to be the inverse of the distance between nodes. Two nodes far apart are unrelated and every node is totally (infinitely) related to itself. Interpreting relatedness as a function of distance has an obvious advantage: computing distances between nodes in a graph is a well studied problem with several known algorithms. After assigning a weight to each arc one can compute the distance as the minimum length of all the paths connecting the two nodes.

On a closer inspection this interpretation of relatedness as the inverse of distance reveals some problems. Consider the graph in Fig. 1. Depending on the weight assigned to the arcs formed by the properties `has type` and `has genre`, the distances between Lady Gaga, Madonna and Queen are the same. If the `has genre` has less weight than `has type`, this would mean that the band Queen is as related to Lady Gaga as Madonna, which obviously should not be the case. On the other hand, if `has type` has less weight than `has genre` then Queen is more related to AC/DC than Lady Gaga or Madonna simply because they are both bands, which also should not be the case.



■ **Figure 1** RDF graph for concepts in music domain.

In the proposed approach we consider *proximity* rather than distance as a measure of relatedness among nodes. By definition⁵, proximity is closeness; the state of being near as in space, time, or relationship. Rather than focusing solely on minimum path length, proximity balances also the number of existing paths between nodes. As a metaphor consider the proximity between two persons. More than resulting from a single common interest, however strong, it results from a collection of common interests.

⁵ <https://en.wiktionary.org/wiki/proximity>

With this notion of proximity, Lady Gaga and Madonna are more related to each other than with Queen since they have two different paths connecting each other, one through `Musical Artist` and another `Pop Rock`. By the same token the band Queen is more related to them than to the band AC/DC.

An algorithm to compute proximity must take into account the several paths connecting 2 nodes and their weights. However, paths are made of several edges, and the weight of an edge should contribute less to proximity as it is further away in the path. In fact, there must be a maximum number of edges in a path, otherwise virtually every node in the graph will be in a path connecting the original nodes.

The proposed algorithm is formalized in Figure 1 using Java syntax⁶. The algorithm uses sets of paths segments catheterized by an intermediary node and its distance to the initial node. Each path segment is an instance of the class defined on the top of Figure 1. The algorithm starts by creating initial sets of path for each of the given terms. These sets have a single path with the node having as label each of the terms and the distance 0.

The algorithm proceeds recursively in successive sets. In each step the given sets of paths segments are intersected, and those found in the intersection form a complete path connecting the original node. They contribute to the proximity with the combined distance to the original nodes, divided by the cube of the step number. This way the longer the longer a path is, the less it contributes to proximity. Since this path is complete these path segments are removed from further processing. After processing the path segments in the intersection, if the maximum number of steps was not reached then each unused path segment is expanded before recursively adding the contribution to the proximity measure of the next step.

Figure 2 shows how the proximity algorithm proceeds to relate the nodes “Madonna” and “Britney Spears”. This examples omits the label nodes and starts with concept nodes associated with the relevant terms. We can see that each node is at the center of a pair of concentric circle. Each circle intersects a set of nodes that are reached from the center with a certain number of path segments. For instance, “Rock Music”, “Musical Artist” and “Pop Music” are all a path segment away from “Madonna”. A similar situation occurs with “Britney Spears” and some nodes are common to both circles, in this case “Musical Artist” and “Pop Music”. These two intermediary nodes contribute with two independent path connecting the original nodes. The remaining nodes, “Rock music” for “Madonna” and “Dance Pop” for “Britney Spears” are used to continue unfolding the sets of nearby nodes connected to the original ones. In this case the node “Music genre” is common to both circles on the second level. This path is longer than the previous ones (i.e. has more path segments) and thus contributes less to proximity. At each level the contribution of new path is much less than those of the previous ones, although they are usually in greater number. After a few levels (typically 5) the algorithms stops.

The proximity algorithm can be extended to relate groups of concepts. This is relevant to relate two web pages, for instance. In this case one can extract are terms in a web pages and consider those that are labels to graph nodes. The proximity between the two node sets can be defined as the average, or the maximum, of all proximity pairs.

3.2 Implementation

The algorithm described in the previous section is implemented by a system called Shakti. This system is responsible for extracting data relevant to a given domain from DBpedia,

⁶ For sake of clarity was used the Java 7 syntax.

■ **Listing 1** The algorithm for computing proximity between nodes.

```

class Path {
    Node node = null;
    int distance = 0;
    // ...
}

public int proximity(String a, String b) {

    Set<Path> pathsA = new HashSet<>();
    Set<Path> pathsB = new HashSet<>();

    pathsA.add(new Path(nodeWithlabel(a), 0));
    pathsB.add(new Path(nodeWithlabel(b), 0));

    return proximity(pathsA, pathsB, 1);
}

private int proximity(Set<Path> pathsA, Set<Path> pathsB, int step) {
    int proximity = 0;

    if (pathsA.isEmpty() || pathsB.isEmpty())
        return proximity;

    Set<Path> unusedA = new HashSet<>(pathsA);
    Set<Path> unusedB = new HashSet<>(pathsB);

    for (Path pathA : pathsA)
        for (Path pathB : pathsB)
            if (pathA.node.equals(pathB.node)) {
                int step3 = step * step * step;
                proximity += (pathA.distance + pathB.distance) / step3;
                unusedA.remove(pathA);
                unusedB.remove(pathB);
            }

    if (step < MAX_STEP) {
        Set<Path> relatedA = related(unusedA);
        Set<Path> relatedB = related(unusedB);

        proximity += proximity(relatedA, relatedB, step + 1);
    }
    return proximity;
}

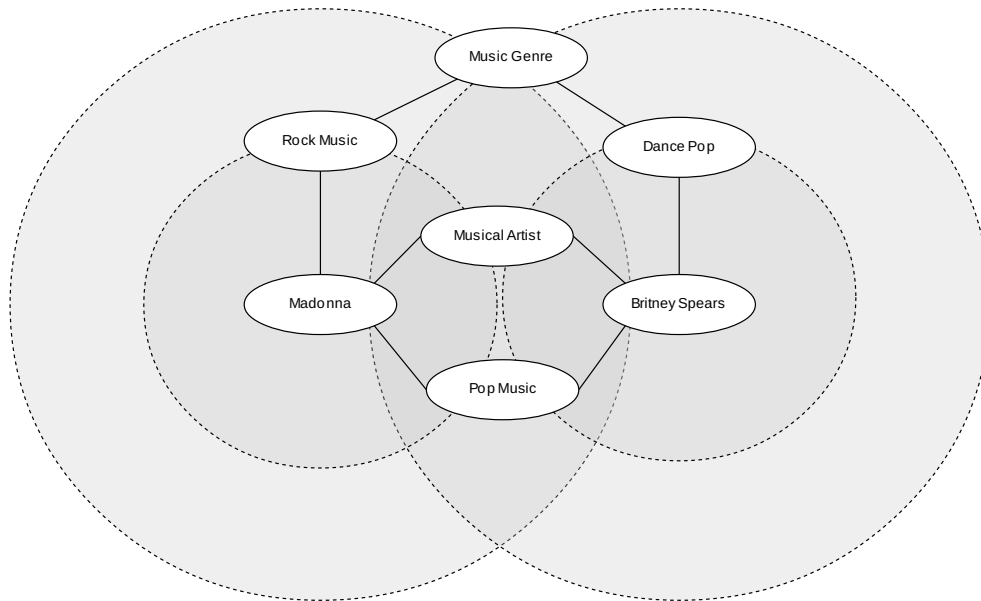
private Set<Path> related(Set<Path> paths) {
    Set<Path> related = new HashSet<>();

    for (Path path: paths)
        for (Arc arc: connectedTo(path.node)) {
            Property property = arc.property;
            Node target = arc.target;

            int distance = path.distance + weightOf(property);

            related.add( new Path(target, distance) );
        }
}

```



■ **Figure 2** Using the proximity algorithm to relate “Madonna” and “Britney Spears.”

to provide a measure of the proximity among concepts in that domain. This system is implemented in Java, an open-source semantic web toolbox called Jena⁷ including application interfaces for RDF and OWL, a SPARQL engine, as well as parsers and serializers for RDF in several formats such as XML, N3 and N-Triples.

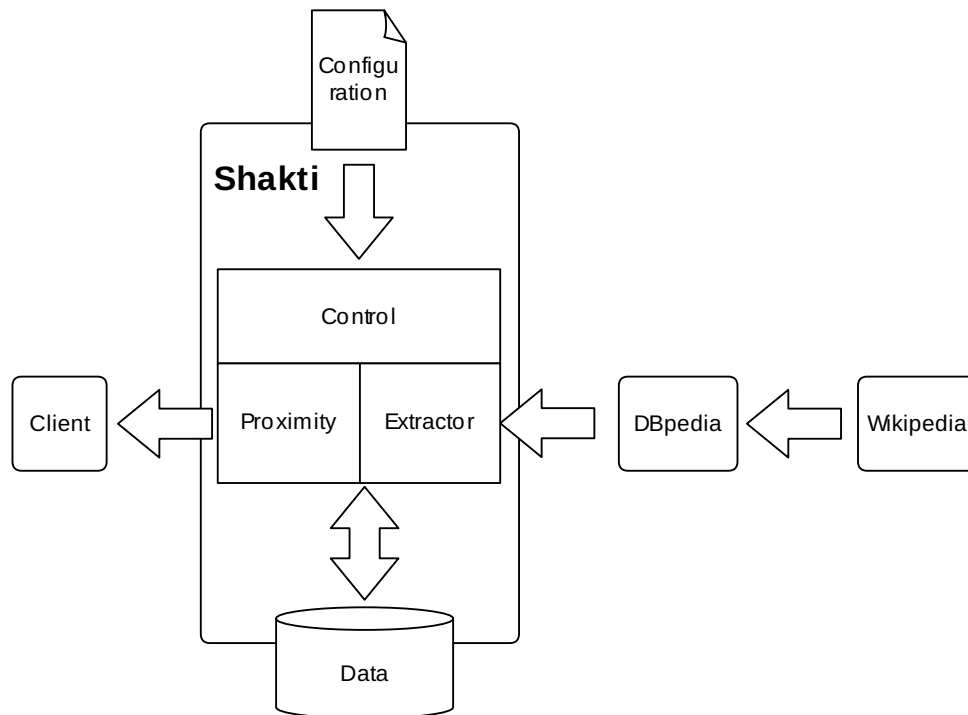
The overall architecture of a Shakti use case is described in the diagram in Figure 3. It shows that Shakti mediates between a client system and DBpedia, that in turn harvests its data from the Wikipedia. The system itself is composed of three main components:

- controller** is parametrized by a configuration file defining a domain and provides control over the other components;
- extractor** fetches data related to a domain from the DBpedia, pre-processes it and stores the graph in a local database;
- proximity** uses local graph to compute the proximity among terms in a pre-configured domain.

The purpose of the controller is twofold: to manage the processes of extracting data and computing proximities by proving configurations to the modules; and to abstract the domain required by client application. For instance, to use Shakti in a music domain it is necessary to identify the relevant classes on concepts, such as *musical artist*, *genre* or *instrument*, as well as the properties that connect them, such as *type*, *has genre* or *plays instrument*. To use Shakti in a different domain, say movies, it is necessary to reconfigure it.

The controller is parametrized by an XML configuration file formally defined by an XML Schema definition as depicted in Figure 4. The top level attributes in this definition configure general parameters, as the URL of the SPARQL endpoint, the natural languages of the labels

⁷ <https://jena.apache.org/>

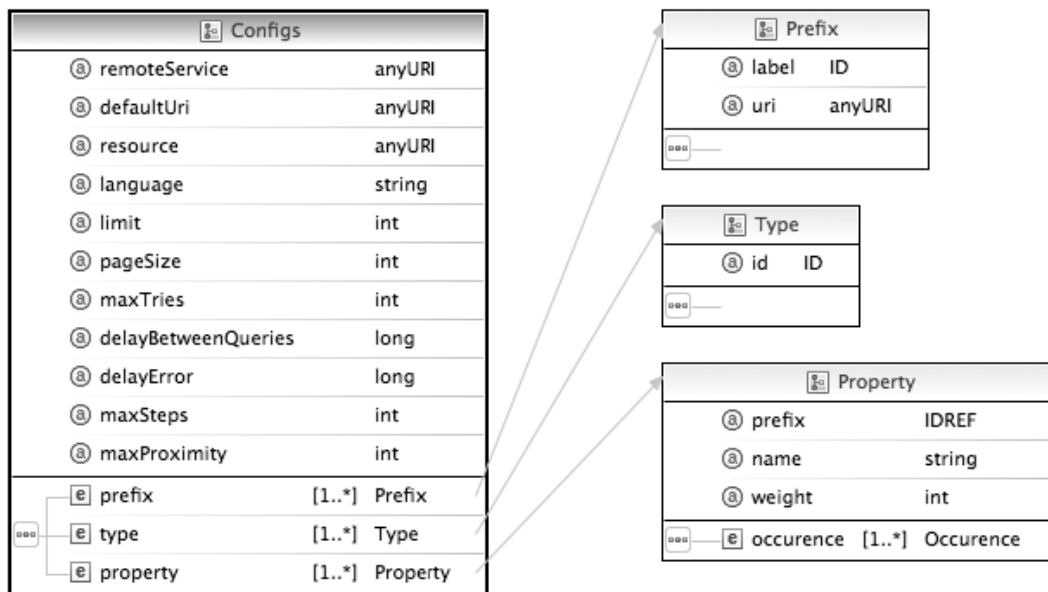


■ **Figure 3** The architecture of Shakti.

(e.g. English, Portuguese), the maximum level used in the proximity algorithm, among others. The top level elements are used for defining prefixes, types and properties. XML prefixes are routinely used in RDF to shorten the URLs used to identify nodes. This configuration enables the declaration of prefixes used in SPARQL queries. The configuration file also enumerates the types (classes) of concepts required by a domain. This ensures that all the concepts with a declared type, having a label in the requested language are downloaded from DBpedia. The declaration of properties has a similar role but it also provides the weights assigned to path segments required by the algorithm. Each property definition includes a set of occurrences since the same name may be used to connect different types. That is, each property occurrence has a domain (source) and a range (target) and these must be one of the previously defined types. These definitions ensure that only the relevant occurrences of a property are effectively fetched from DBpedia.

The *extractor* retrieves data using the SPARQL endpoint of DBpedia. The extractor processes the configuration data provided by the controller and produces SPARQL queries that fetch a DBpedia sub-graph relevant for a given domain. Listing 2 shows an example of a SPARQL query to extract a type declared in the configuration file, where the string “[TYPE]” is replaced by each declared type. Similar queries are used for extracting properties.

Part of the data extracted this way, namely the labels, must be pre-processed. Firstly, multi-word labels are annotated incorrectly with language tags and must be fixed. For instance, a label such as "Lady Gaga@en" must be converted into "Lady Gaga"@en. Secondly all characters between parentheses must be removed. The Wikipedia, and consequently DBpedia, use parentheses to disambiguate concepts when needed. For instance, "Queen



■ **Figure 4** The XML Schema definition of Shakti configuration.

■ **Listing 2** SPARQL query for extracting a type.

```
SELECT ?R ?L
WHERE {
  ?R      rdf:type    dbpedia:[TYPE];
         rdfs:label  ?L.
}
```

(Band)"@en is a different concept from "Queen"@en but in a music setting the term in brackets is not only irrelevant but would disable the identification with the term "Queen" when referring to the actual band. Also, concepts with short labels (less than 3 characters) or solely with digits (e.g. "23") are simply discarded.

The *proximity* module is responsible for computing the relatedness between two terms, or two bags-of-terms, from the graph extracted from DBpedia and already pre-processed. This module maintains a dictionary with all labels in the graph, implemented using a prefix tree, or *trie*. This data structure enables an efficient pre-search of terms, discarding the terms for which relatedness cannot be computed. Following this step, the implementation follows closely the algorithm describes in listing 1.

4 Evaluation

This section presents a use of Skati in the implementation of a recommender developed as part of the project Palco 3.0. This project was targeted to the re-development of an existing Portuguese social network — Palco Principal — whose main subject is alternative music.

The goals of this project include the automatic identification, classification and recommendation of site content. The recommendation service developed for this project is structured around recommenders — pluggable components that generate a recommendation for a certain request based on a given model. Most of the recommenders developed for this service use collaborative filtering. For instance, a typical recommender suggest songs to users in Palco Principal based on the recorded activity of other users. If a user shares a large set

of songs in his or her playlist with other users then it is likely that he or she will enjoy other songs in their playlist.

This approach is very effective and widely used but its main issue is cold start. If the system has no previous record of a new user then it will not be able to produce a recommendation. An alternative is to produce a content-based recommender. To implement such a recommender Shakti was used to find related content on the web site. This recommender can be used on words extracted from the web page itself, such as news articles or interviews, or on tags used to classify web pages, such as musics, photos or videos.

The remainder of this section describes the main steps to define a content recommender for Palco Principal using Shakti and how this experiment was used to evaluate this approach.

4.1 Proximity Based Recommendation

Palco Principal is a music website hence this is the domain that must be used in Shakti. This required selecting DBpedia classes and properties relevant to this domain, preparing DBpedia for extracting data from the Portuguese wikipedia to populate these classes, and configuring Shakti with the relevant types and properties to compute proximities.

DBpedia already has an extensive ontology covering most of the knowledge present in Wikipedia. This is certainly the case with the music domain and all the necessary classes and properties were already available. The DBpedia uses a collection of mapping to extract data present in the info boxes of Wikipedia. Unfortunately these mapping were only available for the English pages of Wikipedia and they had to be adapted for the pages in Portuguese. The DBpedia uses a wiki to maintain these mapping and new mappings of some classes had to be associated with the language label "pt".

In the Shakti it was necessary to configure the XML file to extract the selected classes and properties from DBpedia. These classes, whose mappings were created on DBpedia wiki for Portuguese pages, are:

MusicalArtist solo performers (e.g. Madonna, Sting);

Band groups of musicians performing as a band (e.g. Queen, Bon Jovi);

MusicGenre musical genres (e.g. rock, pop).

The properties associated with these classes that were considered relevant were also inserted in the configuration file and are enumerated in Table 1. This table defines also the weights assigned to properties, with values ranging from 1 to 10, needed for computing proximities. These weights were assigned based on the subjective perception of the authors on the proximity of different bands and artists. A sounder approach to weight calibration was left for future work.

■ **Table 1** Properties.

Property	Type	Classes	Weight
Genre	MusicGenre	Band and MusicalArtist	7
Instrument	Label	Band and MusicalArtist	2
StylisticInfluences	Label	MusicGenre	4
AssociatedBand	Band	Band	10
AssociatedMusicaArtist	MusicalArtist	MusicalArtist	10
CurrentMember	Label	Band	5
PastMember	Label	Band	5

To integrate Shakti with the recommender it was necessary to implement a client application. This application is responsible populating a table with proximities among web pages recorded on the recommender service database. For each page this client application extract a bag-of-words, either the words on the actual page or the tags to classify it. For each pair of bags-of-words it computes a proximity using methods provided by Shakti.

4.2 Results Analysis

Shakti is currently being used in an experimental recommender. Thus, the recommendations are not yet available on the site the Palco Principal. For this reason a comprehensive analysis is not yet possible. This subsection presents some experimental results that are possible to obtain from the current setting.

For this experiment the recommender system computed proximities among news and events pages, which took about a day. In total 57,982 proximities relations among news pages were calculated, plus 59992 among event pages, performing a grand total of 69604805 relations.

■ **Table 2** Proximity between pairs of news pages.

Resource ID	Resource ID	Proximity
3540	2623	0.22
3540	2431	0.21
3540	3000	0.15
3540	4115	0.15
3540	2691	0.15
3540	1892	0.15
3540	2676	0.14
3540	760	0.14
3540	3189	0.14
3540	4397	0.14

Table 2 displays the proximity table for news pages ordered by decreasing proximity. Each id code is a news item in the web site. For this particular entity the recommender searched for content regarding both terms from its text and tags.

To analyze the performance of Shakti the contents of the 2 most related pages — id 3540 (resource A) and id 2623 (resource B) — were compared in detail. The text and tags of this resource can be viewed in Figure 5. In order to calculate proximities, Shakti merge both fields and generates a group of concepts present in the RDF graph. Thus, from all the words of *text* and *tags* fields only the following bag-of-words are actually used to compute proximity: 38 Special, Lynyrd Skynyrd, Bret Michaels. For resource B the bag-of-words considered for computing compute proximity is: Lemmy, Myles Kennedy, Andrew Stockdale, Dave Grohl, Fergie, Ian Astbury, Kid Rock, M. Shadows, Rock, The Sword, Adam Levine, Ozzy Osbourne, Chris Cornell, Duff McKagan, Slash, Iggy Pop. Using these two bags-of-words Shakti computes a proximity of 0.22. The concepts are names of the bands appearing in news text, so the approach of using the this field to determine proximity seems promising.

Analyzing these news items one notices that they are on two musician artists with a musical genre in common, and both playing the guitar. This shows that the two news items are in fact related and a 0.22 proximity seems a reasonable figure. Note that proximities range between 0.0 (unrelated) to 1.0 (the same).



■ **Figure 5** News piece generated from about resource A.

The proximity values computed for all pages vary between 0.1 and 0.22 and the average value of about 0.2. This value is lower than expected. Of course that these figures can be modified simply by reconfiguring the property weights. On the other hand, Shakti determined a non null proximity in 24,401 of a total of 33,616,804 possible relationships, about 0.07%, which is an unsatisfactory figure for a recommendation system.

One of the culprits for these poor results is the text encoding using HTML entities in the database of Palco Principal. For instance, the term "Guns N' Roses" (which is part of the text and tags of resource B) is written in the database in the format "Guns N'Roses". This value is sent to the Shakti. As Shakti is not prepared to receive this type of formatting, it does not detect the word in the dictionary.

Nevertheless, the problems with text encoding alone do not justify the low number recommendations obtained in this experiment. Most probably the sub-ontology extracted from DBpedia does not cover satisfactory the domain of the Palco Principal.

5 Conclusions and Future Work

The goal of the research described in this paper is to measure the relatedness of two terms using the knowledge base of DBpedia. The motivation for this research is to use semantic relatedness in content-based recommenders, in particular in tags provided by users in social networks.

This paper proposes an algorithm to compute the semantic relatedness of two terms as proximity rather than distance, as in similar ontology based approaches. The algorithm ponders the collection on paths connecting the two terms rather using the weights associated to properties on the ontological graph. This algorithm was implemented in a system called Shakti. This system fetches a sub-graph of the ontology in DBpedia relevant to a certain domain and computes the relatedness of terms assigned as labels to concepts. To validate the proposed approach Shakti was used to populated a proximity table on a web recommended service of Palco Principal, a Portuguese social network whose subject is alternative music. The results are promising, although the ontology extracted form DBpedia is not yet covering satisfactory the terms contained on the pages of Palco Principal.

As part of the future work in this research we plan to experiment with larger ontologies, providing better coverage of the underlying domain and validating scalability of Shakti. At this stage most of the effort of using Shakti is configuring this tool. We plan the development of a graphical user interface for assisting the tool users in defining the classes and properties to extract from DBpedia. There are two approaches being considered for this task. On the first approach a seed class is typed in and other related classes and properties in that

domain are suggested for possible inclusion. On the second approach Shakti is fed with a collection of example terms and DBpedia is searched for classes for those terms, as well as relates properties. Independently from the selected approach, the graphical user interface will also assist in the definition of property weights and other general configurations required by Shakti.

Acknowledgments This work is in part funded by the ERDF/COMPETE Programme and by FCT within project FCOMP-01-0124-FEDER-022701. The authors wish also to thank the reviewers for their helpful comments.

References

- 1 Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.
- 2 D. Beckett and B. McBride. Resource description framework (RDF) model and syntax specification (revised). Technical report, W3C, 2004.
- 3 D. Brickley and R.V. Guha. RDF vocabulary description language 1.0: RDF schema. Technical report, W3C, 2004.
- 4 Óscar Corcho and Asunción Gómez-Pérez. A roadmap to ontology specification languages. In *Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling and Management, EKAW '00*, pages 80–96, London, UK, UK, 2000. Springer-Verlag.
- 5 Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407, 1990.
- 6 Christiane Fellbaum, editor. *WordNet An Electronic Lexical Database*. The MIT Press, Cambridge, MA ; London, May 1998.
- 7 Evgeniy Gabrilovich. Feature generation for textual information retrieval using world knowledge. *SIGIR Forum*, 41(2):123–123, December 2007.
- 8 Evgeniy Gabrilovich and Shaul Markovitch. Overcoming the brittleness bottleneck using wikipedia: enhancing text categorization with encyclopedic knowledge. In *proceedings of the 21st national conference on Artificial intelligence - Volume 2, AAAI'06*, pages 1301–1306. AAAI Press, 2006.
- 9 J.J. Jiang and D.W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proc. of the Int'l. Conf. on Research in Computational Linguistics*, pages 19–33, 1997.
- 10 Dekang Lin. An information-theoretic definition of similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning, ICML '98*, pages 296–304, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.
- 11 David Milne and Ian H. Witten. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management, CIKM '08*, pages 509–518, New York, NY, USA, 2008. ACM.
- 12 M. Nilsson, M. Palmer, and J. Brase. The LOM RDF binding - principles and implementation. In *3rd Annual ARIADNE Conference*, 2003.
- 13 Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th international joint conference on Artificial intelligence - Volume 1, IJCAI'95*, pages 448–453, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.
- 14 N. Seco, T. Veale, and J. Hayes. An intrinsic information content metric for semantic similarity in WordNet. *Proc. of ECAI*, 4:1089–1090, 2004.
- 15 I. Smirnov. Overview of stemming algorithms. *Mechanical Translation*, 2008.

- 16 Michael Strube and Simone Paolo Ponzetto. Wikirelate! computing semantic relatedness using wikipedia. In *proceedings of the 21st national conference on Artificial intelligence - Volume 2*, AAAI'06, pages 1419–1424. AAAI Press, 2006.
- 17 Fei Wu and Daniel S. Weld. Automatically refining the wikipedia infobox ontology. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 635–644, New York, NY, USA, 2008. ACM.
- 18 Torsten Zesch and Iryna Gurevych. Automatically creating datasets for measures of semantic relatedness. In *Proceedings of the Workshop on Linguistic Distances*, LD '06, pages 16–24, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

