UDC, DOI: N/A

Using proximity to compute semantic relatedness in RDF graphs

José Paulo Leal

CRACS & INESC-Porto LA, Faculty of Sciences, University of Porto
Porto, Portugal
zp@dcc.fc.up.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 terns in RDF graphs based on the notion of *proximity*. It proposes a formal definition of proximity in terms of the set paths connecting two concept nodes, and an algorithm for finding this set and computing proximity with a given error margin. This algorithm was implemented on a tool called Shakti that extracts relevant ontological data for a given domain from DBpedia – a community effort to extract structured data from the Wikipedia. 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 also reported.

Keywords: semantic similarity, semantic relatedness, ontology generation, web recommendation, processing Wikipedia data

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 is 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 multiword terms in computational linguistics.

The motivation for this research in semantic relatedness comes from another application area, recommendation. Most recommenders use statistical methods, 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 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 or searched 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 items 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 Specification[1], 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. The same would happen with lexical thesaurus, such as WordNet. To create and maintain an ontology adjusted to a very specific kind the best approach is to extract it from an existing source. The DBpedia[2] is a knowledge base that harvests the content of the Wikipedia and thus covers almost all imaginable subjects. It is based on an ontology that classifies Wikipedia pages 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 contributions of this paper is the algorithm used for computing relatedness. Most ontologies based algorithms for computing relatedness assume that ontologies are taxonomies or at least direct acyclic graphs, which is not generally the case of 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.

[1] `http://musicontology.com/`
[2] `http://dbpedia.org/About`

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 introduces the concept of proximity, provides a formal definition of this concept in terms of sets of paths, and presents an algorithm for computing proximity based on the proposed definition. Section 4 presents the design and implementation of Shakti, a tool implementing the proposed algorithm. The following section describes 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.

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 in RDF. Its specification [2] includes a data model and a XML binding. The RDF data model is a collection of triples – subject, predicate and object — that can be viewed as a labeled directed multigraph; 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 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. Wordnet[3]) or Biomedical Informatics (e.g. Gene Ontology[4]). 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] and [11]. This is due to the increasing amount of articles available and the associated semantic information (e.g. article and category links).

[3] http://wordnet.princeton.edu/
[4] http://www.geneontology.org/

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. Proximity

This section presents an approach to compute semantic relatedness using ontological information in RDF graphs. The first subsection provides the motivation for using proximity, rather than distance, as the underlying concept for computing semantic relatedness between two nodes. The following subsection presents a formal definition of the proximity based on sets of paths connecting the nodes. The final subsection outlines the algorithm for computing proximity using the proposed definition.

3.1. Motivation

Concepts on DBPedia are represented by nodes. Take for instance the music domain used for the case study presented in section 5. Singers, bands, music genres, instruments or virtually any concept related to music is represented as a node in DBpedia. These nodes are related by properties, such as `has genre` connecting singers to genres, and thus form a graph. This graph can be retrieved in RDF format using the SPARQL endpoint of DBpedia.

The core idea in the research presented in this paper is to use the RDF graph to compute the similarity between nodes. Actually, the goal is 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`

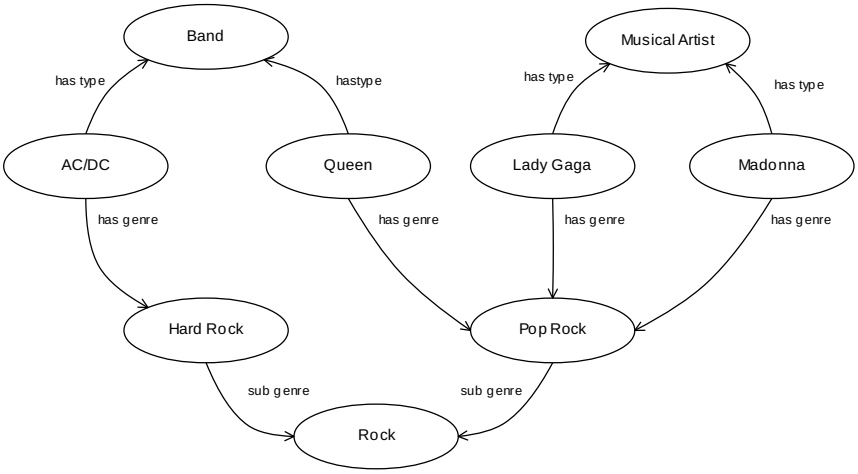

Fig. 1. RDF graph for concepts in music domain

`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.

In the proposed approach we consider *proximity* rather than distance as a measure of relatedness among nodes. By definition[5], 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.

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 two 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 limit in number of edges in a path, as RDF graphs are usually connected graphs.

3.2. Definition

To be of practical use the notion of proximity among RDF nodes needs to be formalized. Proximity must be a function of graph nodes returning their amount

[5] https://en.wiktionary.org/wiki/proximity

of proximity. Given a graph with a set of nodes V the objective of this subsection is thus to define a function

$$p : V \times V \rightarrow [0, 1]$$

The proximity function p must take two nodes and return the "percentage" of proximity between them. That is, proximity of related nodes must be close to 1, with $\forall_{v \in V}\ p(v, v) = 1$, and the proximity of unrelated nodes must be close to 0.

An RDF graph is actually a typed multigraph, meaning that any pair of nodes can be connected by several edges, known as properties. Nodes and specially edges (properties) in RDF graphs have an URI that can be interpreted as a *type*[6]. For the purpose of defining a proximity function only edge types are relevant. Moreover, this approach requires weights associated with edge types, rather then directly to edges as is usual in graph theory.

Consider a direct[7] typed multigraph $G = (V, E, T, W)$ where V is a set of nodes or vertices, E is a set of edges, T is a set of edge types and W is a mapping of types to positive integers. Each edge in E is an ordered triplet (u, v, t) where $u, v \in V$ and $t \in T$.

The set W defines a mapping $w : T \rightarrow \mathbb{N}^+$ and the lower upper bound of weights for all types is

$$\Omega(G) \equiv \max_{t_i \in T}\ w(t_i)$$

The *degree* of a node is the number of edges connecting to it, $deg(u) = \#\{(u', v', t') \in E : u' = u\}$ and the degree of a graph G, denoted $\Delta(G)$, is usually defined as the maximum of the node degrees

$$\Delta(G) = \max_{v \in V}\ deg(v)$$

Given the multigraph G, an *acyclical path* p of size $n \in \mathbb{N}^+$ is defined as a sequence of unrepeated nodes $u_0 \ldots u_n \forall_{0 \leq i,j \leq n} u_i \neq u_j$ connected by edges with type t_i in either direction, that is $\forall_i (u_{i-1}, u_i, t_i) \in E \vee (u_i, u_{i-1}, t_i) \in E$, as follows.

$$p = u_0 \xrightarrow{t_1} u_1 \xrightarrow{t_2} u_2 \ldots u_{n-1} \xrightarrow{t_n} u_n$$

An acyclical path must have at least one edge and cannot have loops. In the remainder of this section an acyclical path is simply referred as a *path*.

The weight function defined above can be extended to paths. The weight of path p is the sum of weights of each edge's type, $w(p) = w(t_1) + w(t_2) + \ldots + w(t_n)$. Since $w(t_i) \leq \Omega(G)$ it results that $w(p) \leq \Omega(G)n$, where n is the size of the path.

The set of all paths connecting vertices u and v with exactly $n \geq 1$ edges is defined as follows.

[6] A type in the usual sense of graph theory, not an RDF Schema or OWL type.

[7] An RDF graph is a direct graph. However, edge direction is irrelevant for the purpose of this definition of proximity.

$$P^n_{u,v} = \{u_0 \xrightarrow{t_1} u_1 \dots u_{n-1} \xrightarrow{t_n} u_n : u = u_o \wedge v = u_n \wedge \forall_{0 \leq i,j \leq n}\, u_i \neq u_j\}$$

The weight of $P^n_{u,v}$ can be computed using the path weight function defined above simply by adding the contribution of each path. The reader should note that $\sum_{p \in P^n_{u,v}} w(p) \leq \Omega(G) n \Delta(G)^n$ since $\forall_{p \in P^n_{u,v}} w(p) \leq \Omega(G) n$ and $\# P^n_{u,v} \leq \Delta(G)^n$.

A proximity function can be defined in terms of these sets of paths. The proximity of a node to itself must be taken as a special case given that $\forall_{n \in \mathbb{N}^+} P^n_{u,u} = \emptyset$. For the general case where the two nodes are different, proximity must take into account each path in $P^n_{u,v}$, for all values of n. However, shorter paths must weight more that longer paths. That is, paths in $P^n_{u,v}$ for smaller values of n must contribute more to proximity than those of larger values of n. Having this in mind the proposed proximity function p is defined as follows.

$$p(u,v) = \begin{cases} 1 & \leftarrow u = v \\ \frac{1}{\Omega(G)} \sum_{n=1}^{\infty} \frac{1}{2^n n \Delta(G)^n} \sum_{p \in P^n_{u,v}} w(p) & \leftarrow u \neq v \end{cases}$$

Since this definition relies on an infinite series one must ensure that it converges. Given that $\sum_{p \in P^n_{u,v}} w(p) \leq \Omega(G) n \Delta(G)^n$, if $u \neq v$, $p(u,v) \leq \sum_{n=1}^{\infty} \frac{1}{2^n} = 1$, and thus the series converges absolutely. It is trivial that the proximity function is non-negative, since all its terms and factors are natural numbers. Hence this also proves that the image of the proposed function is defined within the intended codomain ($[0,1]$).

3.3. Algorithm

The proximity function as defined in the previous subsection requires computing an infinite series. However, since the series defining this function converges absolutely, the first n terms compute the proximity within a known error margin.

The proposed proximity algorithm is formalized in Algorithm 1. It takes a multigraph and two strings, and starts by creating initial sets of paths for each of the given terms. If these sets are equal then proximity is set to its maximum value (1). Otherwise the algorithm computes the sets of paths linking the two nodes with a size under a predefined limit.

To compute the set of paths of size n the algorithm expands half-paths starting on both ends. The set `PathSetA` contains paths starting in the node with label `A` and the set `PathSetB` contains paths ending in the node with label `B`. Paths of size n are those with semi-paths in `PathSetA` and `PathSetB` with a common ending. The contribution of these paths to proximity is computed by summing their weights, using the `PathWeight` function, and divide it by a `denominator` that depends on the value of n. Before proceeding to the next value of n, first the semi-paths from sets `PathSetA` and `PathSetB` are alternately expanded. If both sets were expanded at once only paths with an even number of nodes would be generated.

Algorithm 1: Proximity function

Input : G = (V, E, T, W), A, B
Output: Proximity

$\Omega \leftarrow$ `MaxWeight`(T, W)
$\Delta \leftarrow$ `MaxDegree`(V, E)
PathSetA $\leftarrow$ NodeSetWithLabel (A,V,E,T)
PathSetB $\leftarrow$ NodeSetWithLabel (B,V,E,T)

Proximity $\leftarrow 0$
ExpandLeft $\leftarrow true$

if PathSetA = PathSetB **then**
    Proximity $= 1$
else
    for N $\leftarrow 0$ **to** MaxLevel **do**
        Denominator $\leftarrow 2^{\mathsf{N}}\Omega\mathsf{N}\Delta^{\mathsf{N}}$
                ▷ *Process all paths of size* N
        for PathA $\in$ PathSetA **do**
            for PathB $\in$ PathSetB **do**
                if `LastNodeInPath`(PathA) = `LastNodeInPath`(PathB) **then**
                    Weight $\leftarrow$ `PathWeight`(PathA) + `PathWeight`(PathB)
                    Proximity $\leftarrow$ Proximity + Weight/Denominator
        ▷ *Expand paths one level alternately in each side*
        if ExpandLeft **then**
            PathSetA $\leftarrow$ `ExpandPaths`(PathSetA, E)
        else
            PathSetB $\leftarrow$ `ExpandPaths`(PathSetB, E)
        ExpandLeft $\leftarrow$ `Not`(ExpandLeft)

Function ExpandPaths(PathSet,Edges)

Data: PathSet,Edges
Result: NewPathSet

NewPathSet $\leftarrow \emptyset$
for Path $\in$ PathSet **do**
    LastNode $\leftarrow$ `LastNodeInPath`(Path)
    for NextNode $\in$ {n : (LastNode, n) $\in$ Edges $\vee$ (n, LastNode) $\in$ Edges} **do**
        if NextNode $\notin$ Path **then**
            NewPathSet $\ni$ (Path, NextNode)

The function `ExpandPaths` expands each path in the given set of paths using a set of edges. Paths are expanded at their end with nodes for which there is an edge starting (or ending) at their last node. If the new node already occurs in the selected path then this is not a valid expansion as it would contain a cycle and it is not added to the expanded path set. Note that this function expands the size of the paths rather than the cardinality of the set. The expanded path set contains paths of size $n + 1$ where n is the size of the paths in the original path set. The cardinality of the expanded path set may either increase, decrease, or remain unchanged by this expansion.

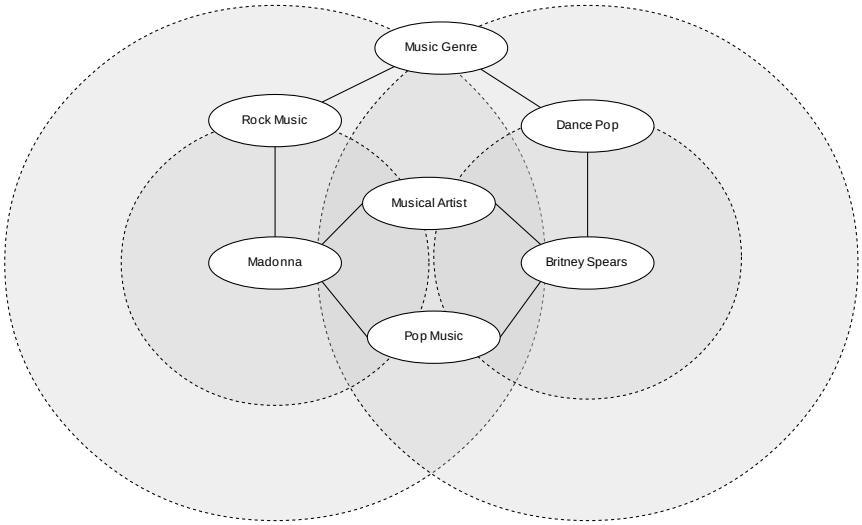

Fig. 2. Using the proximity algorithm to relate “Madonna” and “Britney Spears”

Figure 2 shows how the proximity algorithm proceeds to relate the nodes “Madonna” and “Britney Spears”. This example 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 circles. 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 paths connecting the original modes. 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 paths diminishes, although they are usually in greater number. After a few levels (typically 5) the algorithm stops.

The proximity algorithm can be extended to compare groups of concepts. This is relevant to relate two web pages, for instance. For this purpose a web page is represented by a bag-of-words, where each word occurs in the web page and is also a label of a graph node. The proximity between the two bags-of-words can be defined as the average, or the maximum, of all proximity pairs.

4. Shakti

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, and to provide a measure of the proximity among concepts in that domain. This system is implemented in Java using an open-source semantic web toolbox called Jena[8] 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 preconfigured domain.

The purpose of the controller is twofold: to manage the processes of extracting data and computing proximity values by proving configurations to the modules; and to abstract the domain required by client applications. 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 (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

[8] https://jena.apache.org/

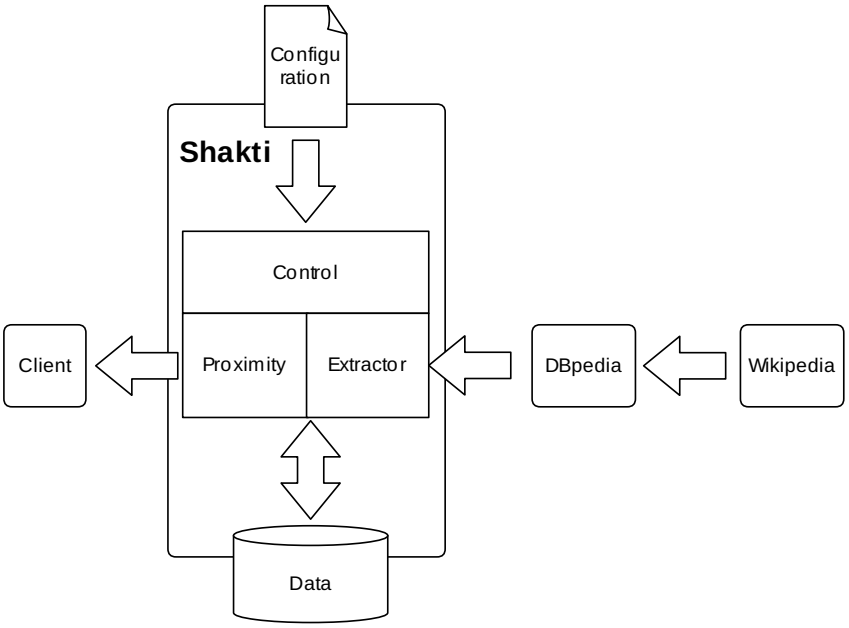

Fig. 3. The architecture of Shakti

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 1.1 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 preprocessed. Firstly, multiword labels are annotated incorrectly with language tags and must

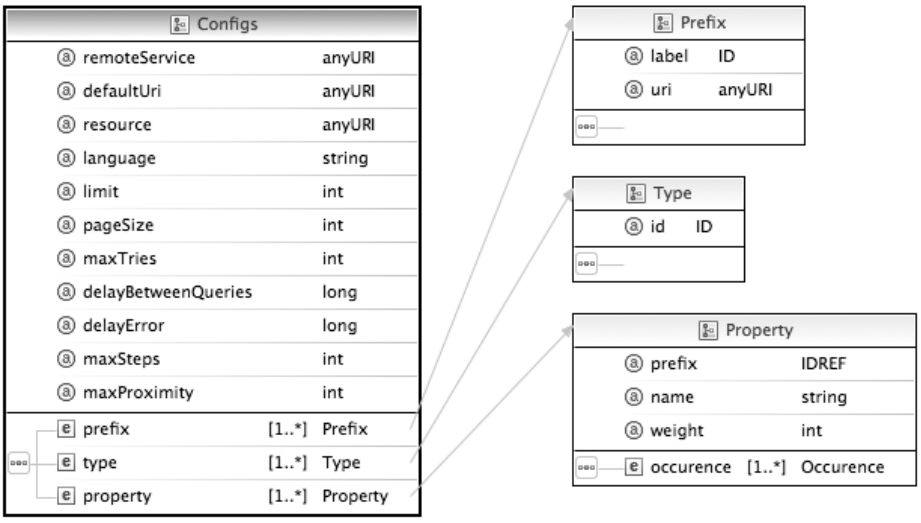

Fig. 4. The XML Schema definition of Shakti configuration

Listing 1.1. SPARQL query for extracting a type

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

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 (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 preprocessed. This module maintains a dictionary with all labels in the graph, implemented using a prefix tree, or *trie*. This data structure enables an efficient screening of terms, discarding those for which relatedness cannot be computed. Following this step, the implementation follows Algorithm 1.

5. 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 of 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.

5.1. 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 proximity values.

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 mappings to extract data present in the info boxes of Wikipedia. Unfortunately these mappings 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 mappings 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 proximity values. 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 of a music domain

Property	**Domain**	**Range**	**Weight**
Genre	Band and MusicalArtist	MusicGenre	7
Instrument	Band and MusicalArtist	*label*	2
StylisticInfluences	MusicGenre	*label*	4
AssociatedBand	Band	Band	10
AssociatedMusicaArtist	MusicalArtist	MusicalArtist	10
CurrentMember	Band	*label*	5
PastMember	Band	*label*	5

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

5.2. Results analysis

Shakti is currently being used in an experimental recommender. Thus, the recommendations are not yet available on the site the of 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 proximity values among news and events pages, which took about a day. In total 57,982 proximity relations among news pages were calculated, plus 59992 among event pages, performing a grand total of 69604805 relations.

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

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

detail. The text and tags of this resource can be viewed in Figure 5. In order to calculate proximity values, 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.

Fig. 5. News piece generated from resource A.

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 proximity values range between 0.0 (unrelated) to 1.0 (the same).

The proximity values computed for all pages vary between 0.1 and 0.22 and the average value of is 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&#039 Roses”. This value is sent to Shakti. As Shakti is not prepared to receive this type of formatting, it does not detect the word in the dictionary.

This experiment suggest that algorithm is producing the expected results. For the pairs of pages that produced a non null proximity the obtained measure is consistent with their degree of relatedness. However, the number of pages that the algorithm was able to relate is insufficient for a recommender system. The problems with text encoding alone do not justify the low number recommendations obtained in this experiment. Most probably the words contained in those pages are not labels in the sub-ontology extracted from DBpedia and it does not not cover satisfactory the domain of Palco Principal. It should be noted that this sub-ontology deals only with artists and bands that have sufficient recognition to have their own entry in Wikipedia. Other concepts related to music, such as musical instruments or music event venues, were not covered. Thus, pages on music festivals featuring garage bands, for instance, or advertising used guitars for sale, would be difficult to relate. In any event, further experimentation is needed to validate both the algorithm itself and the approach of using semantic relatedness a basis for recommendation.

6. 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 BDPedia. 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 proximity, rather than distance, as a means to compute semantic relatedness on RDF nodes. It provides a formal definition of the proximity in terms of the sets of paths connecting the nodes, and an algorithm to determine these sets and compute proximity. The algorithm ponders the collection on paths connecting the two terms 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 populate a proximity table on a web recommender service of Palco Principal, a Portuguese social network whose subject is alternative music. The results are promising, although the ontology extracted from DBpedia is not yet covering satisfactory the terms contained on the pages of Palco Principal.

Part of the future work in this research includes the experimentation 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 related classes and 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.

The validation of the algorithm itself is perhaps the most important part of the future work. It is necessary to compare it experimentally with the results obtained by similar algorithms using standard benchmarks. A testbed for computing similarity and visualizing relatedness among any sets of terms, based on the full DPpedia ontology, is currently being developed. This testbed is expected to be instrumental in the validation of the proposed algorithm.

The fact that the algorithm currently relies on weights being assigned to properties is an obstacle to use it with multiple domains. This issue can be overcome by assigning weights to properties according to their role on the ontology, independently of the domain: `is-a` properties with the maximum weight, `part-of` properties with an intermediary weight, and all other properties with a minimum weight. The testbed will be used to fine-tune these generic weights and to validate this approach to weight assignment.

Acknowledgments

This work is in part funded by the ERDF/COMPETE Programme and by FCT within project FCOMP-01-0124-FEDER-022701. The author wishes to thank to Vânia Rodrigues for her collaboration on the implementation o Shakti, and to Ricardo Queirós, João Delgado and the anonymous reviewers for their helpful comments.

References

1. Baeza-Yates, R.A., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1999)
2. Beckett, D., McBride, B.: Resource description framework (RDF) model and syntax specification (revised). Tech. rep., W3C (2004), http://www.w3.org/TR/REC-rdf-syntax/
3. Brickley, D., Guha, R.: RDF vocabulary description language 1.0: RDF schema. Tech. rep., W3C (2004), http://www.w3.org/TR/rdf-schema/
4. Corcho, O., Gómez-Pérez, A.: A roadmap to ontology specification languages. In: Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling

and Management. pp. 80–96. EKAW '00, Springer-Verlag, London, UK, UK (2000), http://dl.acm.org/citation.cfm?id=645361.650838
5. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. Journal of the American Society for Information Science 41, 391–407 (1990)
6. Fellbaum, C. (ed.): WordNet An Electronic Lexical Database. The MIT Press, Cambridge, MA ; London (May 1998), http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=8106
7. Gabrilovich, E.: Feature generation for textual information retrieval using world knowledge. SIGIR Forum 41(2), 123–123 (Dec 2007), http://doi.acm.org/10.1145/1328964.1328988
8. Gabrilovich, E., Markovitch, S.: Overcoming the brittleness bottleneck using wikipedia: enhancing text categorization with encyclopedic knowledge. In: proceedings of the 21st national conference on Artificial intelligence - Volume 2. pp. 1301–1306. AAAI'06, AAAI Press (2006), http://dl.acm.org/citation.cfm?id=1597348.1597395
9. Jiang, J., Conrath, D.: Semantic similarity based on corpus statistics and lexical taxonomy. In: Proc. of the Int'l. Conf. on Research in Computational Linguistics. pp. 19–33 (1997), http://www.cse.iitb.ac.in/ cs626-449/Papers/WordSimilarity/4.pdf
10. Lin, D.: An information-theoretic definition of similarity. In: Proceedings of the Fifteenth International Conference on Machine Learning. pp. 296–304. ICML '98, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1998), http://dl.acm.org/citation.cfm?id=645527.657297
11. Milne, D., Witten, I.H.: Learning to link with wikipedia. In: Proceedings of the 17th ACM conference on Information and knowledge management. pp. 509–518. CIKM '08, ACM, New York, NY, USA (2008), http://doi.acm.org/10.1145/1458082.1458150
12. Nilsson, M., Palmer, M., Brase, J.: The LOM RDF binding - principles and implementation. In: 3rd Annual ARIADNE Conference (2003)
13. Resnik, P.: Using information content to evaluate semantic similarity in a taxonomy. In: Proceedings of the 14th international joint conference on Artificial intelligence - Volume 1. pp. 448–453. IJCAI'95, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1995), http://dl.acm.org/citation.cfm?id=1625855.1625914
14. Seco, N., Veale, T., Hayes, J.: An intrinsic information content metric for semantic similarity in WordNet. Proc. of ECAI 4, 1089–1090 (2004)
15. Smirnov, I.: Overview of stemming algorithms. Mechanical Translation (2008)
16. Strube, M., Ponzetto, S.P.: Wikirelate! computing semantic relatedness using wikipedia. In: proceedings of the 21st national conference on Artificial intelligence - Volume 2. pp. 1419–1424. AAAI'06, AAAI Press (2006), http://dl.acm.org/citation.cfm?id=1597348.1597414
17. Wu, F., Weld, D.S.: Automatically refining the wikipedia infobox ontology. In: Proceedings of the 17th international conference on World Wide Web. pp. 635–644. WWW '08, ACM, New York, NY, USA (2008), http://doi.acm.org/10.1145/1367497.1367583
18. Zesch, T., Gurevych, I.: Automatically creating datasets for measures of semantic relatedness. In: Proceedings of the Workshop on Linguistic Distances. pp. 16–24. LD '06, Association for Computational Linguistics, Stroudsburg, PA, USA (2006), http://dl.acm.org/citation.cfm?id=1641976.1641980