Computing semantic relatedness using DBPedia

José Paulo Leal, Vânia Rodrigues and Ricardo Queirós

June 22, 2012

Symposium on Languages Applications and Technologies (SLATE) 2012
Outline

Introduction
  Background
  Approach

Algorithm
  Motivation
  Example

Shakti
  Shakti
  Configuration
  Extraction

Evaluation
  Context
  Configuration
  Recommender
  Results

Conclusions
Background

- **Semantic relatedness** in
  - bio-informatics (e.g. gene similarity)
  - geographical informatics (e.g. geographic features)
  - computation linguistics (e.g. multi words terms)
  - web recommendation
Background

- **Semantic relatedness** in
  - bio-informatics (e.g. gene similarity)
  - geographical informatics (e.g. geographic features)
  - computation linguistics (e.g. multi words terms)
  - web recommendation

- **Web recommendation** approaches
  - collaborative filtering
  - content based recommendation
Background

- **Semantic relatedness** in
  - bio-informatics (e.g. gene similarity)
  - geographical informatics (e.g. geographic features)
  - computation linguistics (e.g. multi words terms)
  - **web recommendation**

- **Web recommendation** approaches
  - collaborative filtering
  - content based recommendation

- **Ontologies for content recommendation**
  - creation of ontology adjusted to its application
  - computing ontology-based relatedness
Approach

Compute relatedness from RDF

Computation with Shakti

relatedness as proximity

node proximity algorithm

implemented in Java

configurable to a domain

RDF graphs from DBpedia

Wikipedia's infoboxes

mappings to ontology

persisted in triplestore

public SPARQL endpoint
Approach

Compute relatedness from RDF
Approach

**Compute** relatedness from **RDF**

- **Computation with Shakti**
  - relatedness as proximity
  - node proximity algorithm
  - implemented in Java
  - configurable to a domain
Approach

**Compute relatedness from RDF**

- **Computation with Shakti**
  - relatedness as proximity
  - node proximity algorithm
  - implemented in Java
  - configurable to a domain

- **RDF graphs from DBpedia**
  - Wikipedia’s infoboxes
  - mappings to ontology
  - persisted in triplestore
  - public SPARQL endpoint
Approach

Compute relatedness from RDF

- Computation with Shakti
  - relatedness as proximity
  - node proximity algorithm
  - implemented in Java
  - configurable to a domain

- RDF graphs from DBpedia
  - Wikipedia’s infoboxes
  - mappings to ontology
  - persisted in triplestore
  - public SPARQL endpoint
Motivation

José Paulo Leal, Vânia Rodrigues and Ricardo Queirós

Computing semantic relatedness using DBPedia
Motivation

Compute relatedness from RDF graph assigning weights to arcs
Compute relatedness from RDF graph assigning weights to arcs
Different weight assignments create different distance functions
Motivation

▶ Compute relatedness from RDF graph assigning weights to arcs
▶ Different weight assignments create different distance functions ...
▶ ... but inadequate for proximity (e.g. “has type” vs “has genre”)
Motivation

▸ Compute relatedness from RDF graph assigning weights to arcs
▸ Different weight assignments create different distance functions ...
▸ ... but inadequate for proximity (e.g. “has type” vs “has genre”)
▸ Proximity is not a distance function - lacks triangular inequality
Example

Start with two nodes (e.g. "Madonna" and "Britney Spears")

Find immediate neighbors (e.g. "Rock Music", "Musical Artist")

Compute path contributions for common neighbors (weighted by path length)

Continue recursively with non-common neighbors until max length

Full algorithm in the paper
Example

Start with two nodes labels (e.g. “Madonna” and “Britney Spears”)

José Paulo Leal, Vânia Rodrigues and Ricardo Queirós

Computing semantic relatedness using DBPedia
Example

- Start with two nodes labels (e.g. “Madonna” and “Britney Spears”)
- Find immediate neighbors (e.g. “Rock Music”, “Musical Artist”)
Example

- Start with two nodes labels (e.g. “Madonna” and “Britney Spears”)
- Find immediate neighbors (e.g. “Rock Music”, “Musical Artist”)
- Compute path contributions for common neighbors (weighted by path length)
Example

- Start with two nodes labels (e.g. “Madonna” and “Britney Spears”)
- Find immediate neighbors (e.g. “Rock Music”, “Musical Artist”)
- Compute path contributions for common neighbors (weighted by path length)
- Continue recursively with non-common neighbors until max length
Example

- Start with two nodes labels (e.g. “Madonna” and “Britney Spears”)
- Find immediate neighbors (e.g. “Rock Music”, “Musical Artist”)
- Compute path contributions for common neighbors (weighted by path length)
- Continue recursively with non-common neighbors until max length
Architecture

controller

▶ reads configuration file defining a domain
▶ controls other components

Client

Data

DBpedia

Wikipedia

Extractor

Proximity

Control

Shakti

Configur
ration

José Paulo Leal, Vânia Rodrigues and Ricardo Queirós

Computing semantic relatedness using DBPedia
Architecture

controller  ▶ reads configuration file defining a domain
            ▶ controls other components

eextractor  ▶ fetches data from DBpedia
            ▶ pre-processes and stores it locally
Architecture

controller ▶ reads configuration file defining a domain
▶ controls other components

extractor ▶ fetches data from DBpedia
▶ pre-processes and stores it locally

proximity ▶ uses local RDF graph and domain configuration
▶ implements algorithm, exposing it to clients
Configuration

<table>
<thead>
<tr>
<th>Configs</th>
<th>Prefix</th>
<th>Type</th>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>remoteService</td>
<td>label</td>
<td>prefix</td>
<td>anyURI</td>
</tr>
<tr>
<td>defaultUri</td>
<td>ID</td>
<td>prefix</td>
<td>anyURI</td>
</tr>
<tr>
<td>resource</td>
<td>ID</td>
<td>prefix</td>
<td>anyURI</td>
</tr>
<tr>
<td>language</td>
<td>string</td>
<td>prefix</td>
<td>anyURI</td>
</tr>
<tr>
<td>limit</td>
<td>int</td>
<td>prefix</td>
<td>anyURI</td>
</tr>
<tr>
<td>pageSize</td>
<td>int</td>
<td>prefix</td>
<td>anyURI</td>
</tr>
<tr>
<td>maxTries</td>
<td>int</td>
<td>prefix</td>
<td>anyURI</td>
</tr>
<tr>
<td>delayBetweenQueries</td>
<td>long</td>
<td>prefix</td>
<td>anyURI</td>
</tr>
<tr>
<td>delayError</td>
<td>long</td>
<td>prefix</td>
<td>anyURI</td>
</tr>
<tr>
<td>maxSteps</td>
<td>int</td>
<td>Occurrence</td>
<td>anyURI</td>
</tr>
<tr>
<td>maxProximity</td>
<td>int</td>
<td>Occurrence</td>
<td>anyURI</td>
</tr>
</tbody>
</table>

- Configuration is a valid (XSD) XML document
Configuration

- Configuration is a valid (XSD) XML document
- Defines relevant types (nodes) and properties (arcs)
Configuration

- Configuration is a valid (XSD) XML document
- Defines relevant types (nodes) and properties (arcs)
- Assigns weights to properties occurrences (domain-range)
Configuration

- Configuration is a valid (XSD) XML document
- Defines relevant types (nodes) and properties (arcs)
- Assigns weights to properties occurrences (domain-range)
- Algorithm (maxSteps, MaxProximity) and endpoint configurations
Extraction

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

- Connects with endpoint defined in configuration
- Uses SPARQL templates with defined types and properties
- Hashes and paginates (very) long listings
- Extracts node labels and normalizes them
Extraction

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

- Connects with endpoint defined in configuration
- Uses SPARQL templates with defined types and properties
Extraction

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

- Connects with endpoint defined in configuration
- Uses SPARQL templates with defined types and properties
- Hashes and paginates (very) long listings
Extraction

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

- Connects with endpoint defined in configuration
- Uses SPARQL templates with defined types and properties
- Hashes and paginates (very) long listings
- Extracts node labels and normalizes them
Context

Content based **recommender** for **Palco Principal**
Context

Content based **recommender** for **Palco Principal**

- Palco Principal
  - Portuguese social network on alternative music
  - music recommendation based on machine listening
  - all pages with tags and/or textual content
Content based **recommender** for **Palco Principal**

**Palco Principal**
- Portuguese social network on alternative music
- music recommendation based on machine listening
- all pages with tags and/or textual content

**Recommender**
- implemented on a existing recommendation service
- pluggable components with different approaches
- need for general recommender immune to cold start
DBpedia configuration

- DBpedia proved a wiki for managing ontology and mappings
DBpedia configuration

- DBpedia proved a wiki for managing ontology and mappings
- Ontology contained all the relevant types and properties
DBpedia proved a wiki for managing ontology and mappings

- Ontology contained all the relevant types and properties
- Some types needed Portuguese stringification translations
DBpedia proved a wiki for managing ontology and mappings

- Ontology contained all the relevant types and properties
- Some types needed Portuguese stringification translations
- Mappings for Portuguese Wikipedia were adapted from English
### Shakti configuration

<table>
<thead>
<tr>
<th>Property</th>
<th>Domain</th>
<th>Range</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genre</td>
<td>MusicGenre</td>
<td>Band and MusicalArtist</td>
<td>7</td>
</tr>
<tr>
<td>Instrument</td>
<td>Label</td>
<td>Band and MusicalArtist</td>
<td>2</td>
</tr>
<tr>
<td>StylisticInfluences</td>
<td>Label</td>
<td>MusicGenre</td>
<td>4</td>
</tr>
<tr>
<td>AssociatedBand</td>
<td>Band</td>
<td>Band</td>
<td>10</td>
</tr>
<tr>
<td>AssociatedMusicaArtist</td>
<td>MusicalArtist</td>
<td>MusicalArtist</td>
<td>10</td>
</tr>
<tr>
<td>CurrentMember</td>
<td>Label</td>
<td>Band</td>
<td>5</td>
</tr>
<tr>
<td>PastMember</td>
<td>Label</td>
<td>Band</td>
<td>5</td>
</tr>
</tbody>
</table>

▶ XML configuration file for this domain
### Shakti configuration

<table>
<thead>
<tr>
<th>Property</th>
<th>Domain</th>
<th>Range</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genre</td>
<td>MusicGenre</td>
<td>Band and MusicalArtist</td>
<td>7</td>
</tr>
<tr>
<td>Instrument</td>
<td>Label</td>
<td>Band and MusicalArtist</td>
<td>2</td>
</tr>
<tr>
<td>StylisticInfluences</td>
<td>Label</td>
<td>MusicGenre</td>
<td>4</td>
</tr>
<tr>
<td>AssociatedBand</td>
<td>Band</td>
<td>Band</td>
<td>10</td>
</tr>
<tr>
<td>AssociatedMusicaArtist</td>
<td>MusicalArtist</td>
<td>MusicalArtist</td>
<td>10</td>
</tr>
<tr>
<td>CurrentMember</td>
<td>Label</td>
<td>Band</td>
<td>5</td>
</tr>
<tr>
<td>PastMember</td>
<td>Label</td>
<td>Band</td>
<td>5</td>
</tr>
</tbody>
</table>

- XML configuration file for this domain
- Defines properties with domain and range types
Shakti configuration

<table>
<thead>
<tr>
<th>Property</th>
<th>Domain</th>
<th>Range</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genre</td>
<td>MusicGenre</td>
<td>Band and MusicalArtist</td>
<td>7</td>
</tr>
<tr>
<td>Instrument</td>
<td>Label</td>
<td>Band and MusicalArtist</td>
<td>2</td>
</tr>
<tr>
<td>StylisticInfluences</td>
<td>Label</td>
<td>MusicGenre</td>
<td>4</td>
</tr>
<tr>
<td>AssociatedBand</td>
<td>Band</td>
<td>Band</td>
<td>10</td>
</tr>
<tr>
<td>AssociatedMusicaArtist</td>
<td>MusicalArtist</td>
<td>MusicalArtist</td>
<td>10</td>
</tr>
<tr>
<td>CurrentMember</td>
<td>Label</td>
<td>Band</td>
<td>5</td>
</tr>
<tr>
<td>PastMember</td>
<td>Label</td>
<td>Band</td>
<td>5</td>
</tr>
</tbody>
</table>

- XML configuration file for this domain
- Defines properties with domain and range types
- Ad-hoc definition of weights (sounder calibration needed)
Recommender

<table>
<thead>
<tr>
<th>Resource ID</th>
<th>Resource ID</th>
<th>Proximity</th>
</tr>
</thead>
<tbody>
<tr>
<td>3540</td>
<td>2623</td>
<td>0.22</td>
</tr>
<tr>
<td>3540</td>
<td>2431</td>
<td>0.21</td>
</tr>
<tr>
<td>3540</td>
<td>3000</td>
<td>0.15</td>
</tr>
<tr>
<td>3540</td>
<td>4115</td>
<td>0.15</td>
</tr>
<tr>
<td>3540</td>
<td>2691</td>
<td>0.15</td>
</tr>
<tr>
<td>3540</td>
<td>1892</td>
<td>0.15</td>
</tr>
<tr>
<td>3540</td>
<td>2676</td>
<td>0.14</td>
</tr>
<tr>
<td>3540</td>
<td>760</td>
<td>0.14</td>
</tr>
<tr>
<td>3540</td>
<td>3189</td>
<td>0.14</td>
</tr>
</tbody>
</table>
Recommender

<table>
<thead>
<tr>
<th>Resource ID</th>
<th>Resource ID</th>
<th>Proximity</th>
</tr>
</thead>
<tbody>
<tr>
<td>3540</td>
<td>2623</td>
<td>0.22</td>
</tr>
<tr>
<td>3540</td>
<td>2431</td>
<td>0.21</td>
</tr>
<tr>
<td>3540</td>
<td>3000</td>
<td>0.15</td>
</tr>
<tr>
<td>3540</td>
<td>4115</td>
<td>0.15</td>
</tr>
<tr>
<td>3540</td>
<td>2691</td>
<td>0.15</td>
</tr>
<tr>
<td>3540</td>
<td>1892</td>
<td>0.15</td>
</tr>
<tr>
<td>3540</td>
<td>2676</td>
<td>0.14</td>
</tr>
<tr>
<td>3540</td>
<td>760</td>
<td>0.14</td>
</tr>
<tr>
<td>3540</td>
<td>3189</td>
<td>0.14</td>
</tr>
</tbody>
</table>

- Base on recommendation service networked with Palco Principal
Recommender

<table>
<thead>
<tr>
<th>Resource ID</th>
<th>Resource ID</th>
<th>Proximity</th>
</tr>
</thead>
<tbody>
<tr>
<td>3540</td>
<td>2623</td>
<td>0.22</td>
</tr>
<tr>
<td>3540</td>
<td>2431</td>
<td>0.21</td>
</tr>
<tr>
<td>3540</td>
<td>3000</td>
<td>0.15</td>
</tr>
<tr>
<td>3540</td>
<td>4115</td>
<td>0.15</td>
</tr>
<tr>
<td>3540</td>
<td>2691</td>
<td>0.15</td>
</tr>
<tr>
<td>3540</td>
<td>1892</td>
<td>0.15</td>
</tr>
<tr>
<td>3540</td>
<td>2676</td>
<td>0.14</td>
</tr>
<tr>
<td>3540</td>
<td>760</td>
<td>0.14</td>
</tr>
<tr>
<td>3540</td>
<td>3189</td>
<td>0.14</td>
</tr>
</tbody>
</table>

- Base on recommendation service networked with Palco Principal
- Provides Java interface to the API of Palco Principal
# Recommender

<table>
<thead>
<tr>
<th>Resource ID</th>
<th>Resource ID</th>
<th>Proximity</th>
</tr>
</thead>
<tbody>
<tr>
<td>3540</td>
<td>2623</td>
<td>0.22</td>
</tr>
<tr>
<td>3540</td>
<td>2431</td>
<td>0.21</td>
</tr>
<tr>
<td>3540</td>
<td>3000</td>
<td>0.15</td>
</tr>
<tr>
<td>3540</td>
<td>4115</td>
<td>0.15</td>
</tr>
<tr>
<td>3540</td>
<td>2691</td>
<td>0.15</td>
</tr>
<tr>
<td>3540</td>
<td>1892</td>
<td>0.15</td>
</tr>
<tr>
<td>3540</td>
<td>2676</td>
<td>0.14</td>
</tr>
<tr>
<td>3540</td>
<td>760</td>
<td>0.14</td>
</tr>
<tr>
<td>3540</td>
<td>3189</td>
<td>0.14</td>
</tr>
</tbody>
</table>

- Base on recommendation service networked with Palco Principal
- Provides Java interface to the API of Palco Principal
- Expects a proximity matrix for a set of items
Recommender

<table>
<thead>
<tr>
<th>Resource ID</th>
<th>Resource ID</th>
<th>Proximity</th>
</tr>
</thead>
<tbody>
<tr>
<td>3540</td>
<td>2623</td>
<td>0.22</td>
</tr>
<tr>
<td>3540</td>
<td>2431</td>
<td>0.21</td>
</tr>
<tr>
<td>3540</td>
<td>3000</td>
<td>0.15</td>
</tr>
<tr>
<td>3540</td>
<td>4115</td>
<td>0.15</td>
</tr>
<tr>
<td>3540</td>
<td>2691</td>
<td>0.15</td>
</tr>
<tr>
<td>3540</td>
<td>1892</td>
<td>0.15</td>
</tr>
<tr>
<td>3540</td>
<td>2676</td>
<td>0.14</td>
</tr>
<tr>
<td>3540</td>
<td>760</td>
<td>0.14</td>
</tr>
<tr>
<td>3540</td>
<td>3189</td>
<td>0.14</td>
</tr>
</tbody>
</table>

- Base on recommendation service networked with Palco Principal
- Provides Java interface to the API of Palco Principal
- Expects a proximity matrix for a set of items
- Binding code to read API, use Shakti and populate proximity matrix
Results

Novo álbum de Bret Michaels chega hoje às lojas americanas

2010-07-06 12:22 inserido por Melanie Antunes

O novo álbum do norte-americano Bret Michaels chega hoje às lojas dos Estados Unidos. “Custom Built” conta com a participação da jovem cantora Miley Cirus, também conhecida como Hannah Montana, no tema Nothin’ To Lose.
Computed proximities for 33,616,808 possible relations
Results

- Computed proximities for 33,616,808 possible relations
- Resulting values ranging from 0.1 to 0.22 (max 1.0)
Results

- Computed proximities for 33,616,808 possible relations
- Resulting values ranging from 0.1 to 0.22 (max 1.0)
- Average proximity of 0.2 (Null proximities discarded)
Results

- Computed proximities for 33,616,808 possible relations
- Resulting values ranging from 0.1 to 0.22 (max 1.0)
- Average proximity of 0.2 (Null proximities discarded)
- Non null proximities in just 24,401 cases (0.07%)
Results

- Computed proximities for 33,616,808 possible relations
- Resulting values ranging from 0.1 to 0.22 (max 1.0)
- Average proximity of 0.2 (Null proximities discarded)
- Non null proximities in just 24,401 cases (0.07%)
- Tested ontology does not cover most tags and text
Conclusions
Conclusions

Contributions

- algorithm for computing semantic relatedness from RDF graphs
- system for extracting RDF from DBpedia and computing proximity
- evaluation in recommender system

Future work

- Test Shakti in new domains (e.g., films)
- use larger ontologies providing better coverage
- automatic weight calibration
Conclusions

Contributions
- algorithm for computing semantic relatedness from RDF graphs
- system for extracting RDF from DBpedia and computing proximity
- evaluation in recommender system

Future work
- Test Shakti in new domains (e.g. films)
- use larger ontologies providing better coverage
- automatic weight calibration
Thanks for your attention

José Paulo Leal
zp@dcc.fc.up.pt