CeADAR – Centre for Applied Data Analytics Research

Enterprise Ireland Data Analytics Technology Centre

OntoCore - Technical Specification

Document Type:	Technical Spec
Project Title:	CeADAR
DDN Theme:	1 – Intelligent Analytic Interfaces
Theme Leader:	Sarah Jane Delany (DIT)
DDN Sub-Theme:	1.1 – Ease of Interaction
Authors:	Eoghan O'Shea & Robert Ross
Document Version:	1.0
Date of Delivery to ISG:	18th Sept 2015
Number of pages:	7

ABSTRACT
In this report we provide a brief Technical Specification of the OntoCore system, to be developed as part of the CeADAR Ease of Interaction theme. The OntoCore system will use Named Entity Recognition (NER) to identify basic terminology (keywords) from unstructured content, enrich this with Linked Open Data (LOD), and then relate this enriched terminology back to the original content as metadata.

Contents

1 Description of Industry Needs **3**

2 System(s) Involved **3**

3 Approach **4**
3.1 Named Entity Recognition 4
3.2 Annotating Text with Linked Open Data 5
3.2.1 PageRank 5
3.2.2 Context Similarity 6

4 Risks/Challenges of this/these approaches **6**

1 Description of Industry Needs

From conversations with our industry partners we have identified a number of typical use-cases for the OntoCore system. We will illustrate these use-cases with simple examples.

Suppose our business user is in the cloud computing area and they want to identify potential customers in order to offer them their services. To do this, they need to first identify all keywords relating to cloud computing. The challenge then is to take the single keyword "cloud computing" and annotate it with additional information relating to this area. Using Linked Open Data (LOD), all relevant terms relating to "cloud computing", i.e. metadata relating to this term, can be recovered, e.g. terms like DaaS, SaaS, Amazon AWS, etc. By searching for these recovered terms on social media, job posts or any other relevant unstructured data, businesses/people who have a potential need for cloud computing services can then be identified and targeted.

In a company that deals with branded goods and products, taking the name of the branded product and enriching it with LOD would allow a company to recover metadata relating to that product. This metadata would then allow a company to automatically categorise ('basket') the product within a certain broader product area.

Many businesses have "tradestyles" i.e. Doing Business As, or Trading As, which are associated with the legal name. By taking the business' tradestyle as the input keyword, OntoCore would enable a company to link this tradestyle with any corresponding legal names present within the LOD. That is, in OntoCore, the original tradestyle keyword would be enriched with all associated terms present in the LOD, including the legal name, if present.

On other occasions it is likely that more than single keywords need to be annotated. For example, taking a piece of unstructured content, a company may want to extract all the relevant "named entities" or keywords in that content, to enrich these using LOD and, by so doing, produce metadata. Using this metadata, the company could identify the most relevant domain the unstructured content relates to, allowing them to potentially target advertisements at the creators of this unstructured data.

Doing something similar with all the unstructured or semi-structured data that a company itself possesses would allow a company to automatically create and populate their own domain ontology. This would enable a company to better leverage their own data, e.g. by identifying concepts in their data and the relationships between these concepts, potentially allowing a company to spot new product opportunities.

This document provides a high-level technical specification of the planned OntoCore system.

2 System(s) Involved

Where required, unstructured sample data for this demonstrator will be obtained from the web. Data from different domains, e.g. computing, business, etc. can be tested.

OntoCore will be a web-based interface that will allow a user to input unstructured data via a search box (e.g. single keywords, short phrases) or by uploading plain text (.txt) format files (e.g. for larger pieces of unstructured text obtained from the web). OntoCore will not be designed to perform any form of data-type conversion.

For Named Entity Recognition, the extraction of relevant keywords from the text (a significant component of this project), we will use the Named Entity Recognition component of the NLTK framework[1].

The Linked Open Data (LOD) used in this work will come from DBpedia[2], which has been built up through an automated analysis of Wikipedia articles to create an open database based on RDF technology.

The main analysis, including annotating text with LOD, will be performed by Python[3] processes. A WSGI-based[4] web frontend interface running JQuery[5] and Bootstrap[6] will be provided to input the unstructured data, set up the analysis, recover a list of metadata (enriched keywords), and allow a basic visualisation of the result in a web browser.

Different techniques for visualising this data will be explored. The system will allow output of the enriched keywords in a structured format (OWL[7]/RDF[8]), for use in the creation of a useful domain ontology.

3 Approach

The approach we will employ is as follows:

– A user will enter their text (typed-in or a part of an uploaded .txt file);
– The system activates and performs Named Entity Recognition (NER) to extract all relevant keywords from the text;
– The system enriches each of the extracted keywords with additional terminology obtained by reference to a LOD dataset (annotating the text);
– The enriched terminology found for each keyword will be displayed on screen, either as text or in a visualisation;
– In the last step the system will allow the user to output the enriched keywords in a structured format (OWL/RDF) for loading into an ontology.

This approach has two key elements which we discuss more fully below;

- Named Entity Recognition (NER)
- Annotating Text with Linked Open Data

3.1 Named Entity Recognition

In order to carry out Named Entity Recognition (NER), we will use the NER component of the NLTK framework which has been developed in the Python programming language. NLTK provides an interface that allows the use of the Stanford NER within it, if required. There are

[1]`http://www.nltk.org/`
[2]`http://wiki.dbpedia.org/`
[3]`https://www.python.org/`
[4]`http://wsgi.readthedocs.org/en/latest/`
[5]`https://jquery.com`
[6]`http://getbootstrap.com`
[7]`http://www.w3.org/TR/owl-features/`
[8]`http://www.w3.org/RDF`

two approaches that can be taken in carrying out the task of NER: (i) chunking with regular expressions; (ii) training a chunk parser.

In order to train a chunk parser, we will require access to a relevant corpus for training, in the chosen domain. Depending on the domain chosen, we may have to label training data by hand and build up a training corpus in this way. The need to train the NER for particular domains will limit the range of this proof-of-concept demonstrator. A "chunker" using regular expressions may be the more robust approach to take, allowing the proof-of-concept demonstrator to be domain agnostic.

We note that chunking with regular expressions has the advantage that it gives more control over the tag patterns we might want to match, while being difficult to come up with a set of rules that will capture, e.g. all noun phrases (NP). Chunking by training a chunk parser is potentially more accurate but there is likely to be a need to do post-processing to filter unwanted words.

As a first step, a simplified chunker using a regular expression that just extracts all NP entities from the text will be explored.

In broad terms, the approach taken to carry out NER would involve the input of a plain text file, splitting the text up through tokenization, passing the result through a Parts of Speech (POS) tagger, followed by chunking and finally keyword extraction and output.

3.2 Annotating Text with Linked Open Data

The task of annotating text with LOD is as follows: given the keywords extracted from NER the task is to identify the relevant LOD information (resources) that best matches the keyword in its original context.

As mentioned above, we will use DBpedia in this project, which allows us to take advantage of its RDF structure, based on relations between the resources, and the human readable description of a resource generally found under *rdfs:comment*. We will access the DBpedia database using SPARQL[9], making use of a number of packages in Python for this purpose.

We will follow Rusu et al. (2011) [1] here. Rusu et al. explored using Page Rank and Context Similarity algorithms to find the most relevant LOD information.

3.2.1 PageRank

LOD datasets exhibit a graphical structure based on a relationship between resources, e.g. between an instance and a class described by *rdf: type* or between a class and its superclass described by *rdfs:subClassOf*. In order to apply the PageRank algorithm in the case of an LOD dataset there are a number of steps to be followed:

- Build a graph of the LOD dataset where vertices represent all the LOD dataset resources and the edges the relationships between these resources, e.g. using *igraph*[10];

[9] `http://www.w3.org/TR/rdf-sparql-query/`
[10] `http://igraph.org/`

- Take the extracted keyword, w_a, to be annotated, and search within DBpedia, e.g. using SPARQL, for all related terms (candidate resources). We consider as a candidate resource for w_a, all the resources defined by the LOD dataset with w_a as their $rdfs{:}label$;
- Employ an initialisation step, where all the graph vertices are set to 0, if the vertex does not represent one of the candidate resources found in the previous step, or 1/R, with R being the total number of candidate resources found. These values are the initial values to be used in PageRank;
- Compute the PageRank value for each vertex using the formula:

$$PR[V_i] = \frac{1-d}{N} + D. \sum_{V_j \in M(V_i)} \frac{PR[V_j]}{L(V_j)} \tag{1}$$

  where $PR[V_i]$ is the PageRank for each vertex, i, N is the total number of vertices in the graph, D is the damping factor (typically set to 0.85), $M(V_i)$ is the number of vertices ('pages') that link (inbound) to V_i and $L(V_j)$ is the number of outbound links on vertex j;
- When the PageRank values converge below a threshold value (to be determined), the candidate resource with the highest PageRank score is selected for the extracted keyword, w_a.

3.2.2 Context Similarity

An alternative to PageRank is to use what Rusu et al. call Context Similarity [the following is taken loosely from their text]. Context Similarity makes use of the human-readable description of a resource found under *rdfs:comment* in DBpedia. In this technique, each candidate resource is essentially scored based on the word overlap between the context around the keyword w_a and the human-readable descriptions for a candidate resource found in *rdfs:comment* (as before, we choose as a candidate resource all the resources defined by the LOD dataset with w_a as their *rdfs:label*). The candidate resource with the highest score for each word, w_a, will be selected as the annotation (metadata). The context for w_a is represented by the surrounding words in the text, e.g. all words from the same sentence or paragraph. The overlap between a candidate resource and the word context is computed using cosine similarity defined as:

$$sim_{cos}(A, B) = \frac{A.B}{\|A\|\|B\|} \tag{2}$$

where A and B are two bag-of-words vectors.

4 Risks/Challenges of this/these approaches

For the NER part of the project, we will investigate the use of regular expressions and the training of a chunk parser. The task of training a chunk parser will require a labelled corpus for each particular domain. This is a challenging task to complete in the time available and it is likely that we will have to confine ourselves to the use of regular expressions for chunking.

The PageRank algorithm will be the preferred solution to this problem. However this requires the creation of graphs from LOD. Due to the size of DBpedia, converting all of it to a usable graph for this work may be challenging. The alternative is to use the "Context Similarity" approach as discussed above. In either case, we will likely need to confine our initial research to specific domains.

References

[1] Delia Rusu, Blaz Fortuna, and Dunja Mladenic. Automatically annotating text with linked open data. In *In 4th Linked Data on the Web Workshop (LDOW 2011), 20th World Wide Web Conference*, 2011.