Using DBPedia to bootstrap new Linked Data

Alexiei Dingli
Department of Intelligent Computer Systems
Faculty of ICT
University of Malta
Msida MSD 2080, Malta
alexiei.dingli@um.edu.mt

Silvio Abela
Department of Intelligent Computer Systems
Faculty of ICT
University of Malta
Msida MSD 2080, Malta
sabe0004@um.edu.mt

***Abstract*—If the documents on the WWW were somehow structured, then machines can be made to extract meaning (or semantics) from the content and help us find more data that is relevant to what we search. There is an effort to find better ways to include machine-meaning in the documents already present on the WWW by using Natural Language Processing (NLP) techniques and Web technologies such as XML and RDF that are used to insert and represent the "meaning" in the extracted content. We propose an application that uses Information Extraction to extract patterns from a human readable text and use it to try and find similar patterns elsewhere by searching the WWW. This is done to bootstrap the creation of further data elements. These data elements are then stored in RDF format and reused in other searches. Our evaluation show that this approach gives an encouraging degree of success with a precision of 79% and a recall of 71%.**

Keywords- RDF; Linked Data; Semantic Web; XML

I. INTRODUCTION

The WWW has become a powerful modern medium that in some ways is surpassing the old-style media. This is true of advertising where in the United Kingdom online advertising has surpassed that of television [1] and in the United States, in 2010, a total of $12 billion was spent on Web advertising [2]. This does not make the Web a structured medium, almost totally the opposite. [3] dreamt of a web where humans and machines share the contents and help each other in doing so. He said everyone should publish **linked data** that automatically links pieces of data together [4]. Thus, the aim of this research will be to increase Linked Data triples by using patterns extracted from parts of text to find similar patterns and generate new triples by using Information Extraction techniques.

II. BACKGROUND

Our research will be divided in two: ***Linked Data*** and ***Information Extraction***. We will show that there is a link between the two so information islands are linked together more.

A. Publishing Linked Data

With more linked data available we will be able to query the Web as a database [5]. The Web of Data will use names for ***things***. A book, a person, a car, a cat; they are all "things" having a URI as a name. To be able to publish Linked Data, Tim Berners-Lee listed the following four principles [6]:

1) Use URIs as names for things.
2) Use HTTP URIs so that people can look up those names.
3) When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL).
4) Include links to other URIs so that they can discover more things.

Since the beginning of the Linked Data project (http://linkeddata.org/) the number of datasets published as linked data now stands at 203 from a humble 12 in 2007 [7]. One of the first entities to use linked data was the British Broadcasting Corporation (BBC) that used linked data for all its programs in an effort to centralise the vast amount of information from all its programs micro sites [8].

B. RDF and DBPedia

The Resource Description Framework (RDF) is a simple graph model of the form ***subject-predicate- object*** hence triple. It was developed to describe web resources and machine readable information [5]. The graph has two nodes that may be either blank or a URI with a directed arc (the predicate) *always* a URI.
DBPedia is an effort to extract information from Wikipedia articles that have info boxes although the number of such articles is roughly a third of all the English articles [9]. The effort produced an astonishing 25 billion triples (http://www4.wiwiss.fu-berlin.de/lodcloud) but this is only a drop when compared with the vast amounts of information on the indexed Web (http://www.worldwidewebsize.com). The main difficulty to extract data from information on the WWW is that the latter is inconsistent, ambiguous, uncertain and whose corpus is constantly changing [10].

C. Natural Language Processing

This is the study of text processing within a computer system for a spoken or written language [11]. It touches on three main areas of understanding: Theoretical Linguistics, Computational Linguistics and Articial Intelligence with the greatest advances coming with the computer age. In 1950s Alan Turing hinted that machine translation needed to be unambiguous and that machines need to learn to *think* [12]. During the 1960's and early 1970's, ELIZA [13] and SHRDLU [14], were an early attempt at NLP and Artificial Intelligence (AI). Although from then, technology has mushroomed we are still far from an ideal situation due to the many nuances in spoken and written languages.

D. *Extracting Information from the Web*

Before we extract information we need to retrieve it. With Information Retrieval (IR) we get a subset of documents from a collection (the corpus) whilst with Information Extraction (IE) we extract facts or structured information from those documents [15]. IE is used in another field called Named Entity Recognition (NER) where special tools identify different types of semantics from words such as names, nouns, etc. NER tools determine what we take foregranted while reading such as paragraphs or sentence endings [11]. NER tools contain resources such as Tokenisers, Part of Speech taggers (POS), Sentence Parsers and Semantic Taggers. These aid the system to successfully process a body of text.

III. METHODOLOGY

We aim to extract patterns from text and find similar patterns from the Web using GATE (http://gate.ac.uk) and JENA (http://jena.sourceforge.net).

The Gate system encompasses a host of well written tools providing a solid base for IE, and NER. It also boasts its own complete IE system called ANNIE (a Nearly New Information Extraction system). ANNIE makes use of JAPE (Java Annotations Pattern Engine). Gate uses

- Features: attribute/value pairs
- Corpora: collection(s) of document
- Documents: the input documents
- Annotations: directed acyclic graphs modelled as annotations using XML

We used ANNIE's resources to process the input into sentences, nouns, verbs etc. ANNIE does this by using JAPE which provides finite state transduction over the annotations based on regular expressions [16]. ANNIE sets a serial pipeline of text processing resources that inserts a unique ID, type and offsets within the text. Feature map.

By using JAPE grammars, ANNIE matches wanted patterns and then inserts these newly matched patterns into new feature maps. A JAPE grammar consists of a left hand side, containing annotation patterns, and a right hand side containing Java code to manipulate those patterns.

We also make use of Jena, a framework for reading and writing RDF data as XML. Jena also provides a backend triple database that can be either accessed from command

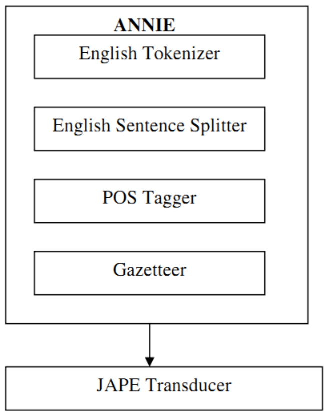

Figure 1. A typical ANNIE pipeline - source The GATE user manual

line or from a web interface called Fuseki. SPARQL is a query language for expressing RDF queries over triple-stores (http://www.w3.org). A SPARQL query may contain triples, conjunctions or disjunctions and some other optional patterns. Its syntax is similar to SQL with queries starting with a **SELECT** statement.

A. *Overview*

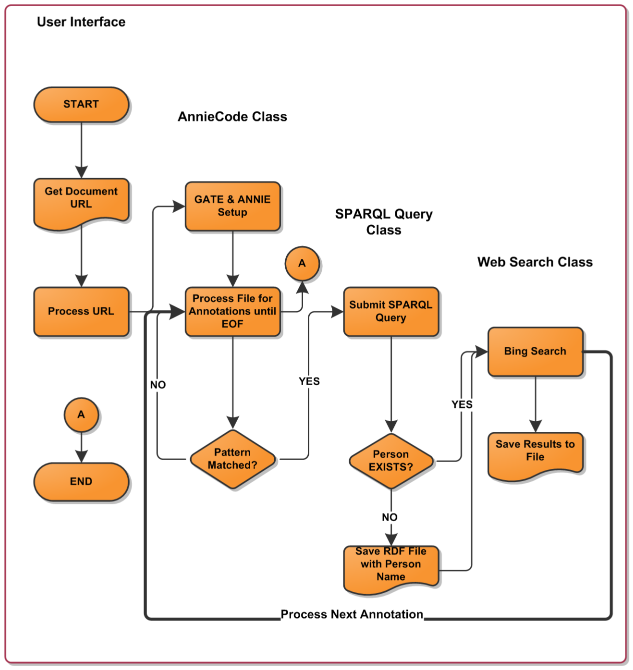

Figure 2. High-level System Flowchart

We propose a system that processes input from text documents to find patterns by using Natural Language Processing techniques. We use SPARQL queries over a database containing over 1.7 million triples and submit Web queries to find other similar patterns on the open WWW. To aid our system to find the required patterns we will also propose

two new JAPE grammars that match the patterns we need in the text input.
Our system presents a simple user interface that processes input documents using four stages:

1) The input stage whereby the system is given an initial URL of a document such as http://en.wikipedia.org/wiki/Barack_Obama which is then downloaded and stored in the system for future processing.

2) The processing stage where GATE is used to annotate text with specific rules or patterns. An example rule might be ...

   Locate the sentence which contains a date, a location, a person and a born event

   A date, location and a person are standard extraction patterns found in Gate. A born event is an additional pattern which we crafted to identify phrases such as ”born on”, etc.

   The rule eventually extracts sentences such as **”Obama was born on August 4, 1961, in Hawaii”** since **Obama** will be recognised as a peron, **”was born on”** is recognised by the born event extraction patterns, **August 4,1961** is a date and **Hawaii** is a location.

3) In the querying stage, we query our database to check if the data we just extracted exists in the database. If it doesn’t, it is inserted in the database as a new RDF data item. An issue to consider is the validity of the data since different pages might return different results for the same person. In the example we’re considering, we found that almost 2 million pages claim that **”Obama was born in Kenya”** and not in Hawaii. After removing the reposted articles and comparing the top patterns extracted, only one piece of data will prevail and that data will be inserted in the database. This is definitely not a foolproof way of ascertaining the truth however, it returns good results in most cases.

4) The Web search stage looks for patterns, similar to the ones in the database in order to extract new information. By similar we mean, that the data just retrieved is sent to the search engine but one of the patterns is omitted. This will retrieve other variances which might not have been covered. So using the data we just retrieved about Obama but omitting the born event, we discover a new piece of data such as

   ”Obama’s date of birth is August 4, 1961, in Hawaii”

   We can see that this data is very similar since all the data items match however the born event pattern is very different. Thus, we use this phrase to construct new patterns by adding wildcards such as

   ***’s date of birth is *, in ***

   This new query will then be fed into Bing and new pages are retrieved which are then sent to the input stage mentioned earlier and the cycle is repeated. By using this simple bootstrapping approach, new information can be discovered all over the web thus generating new linked data almost automatically.

IV. Testing

We tested our application with a small text file containing five seed samples that contain the four pattern parts we need to match in no particular order and in one complete short sentence.

- PersonEvent - *Kenny Matthieson Dalglish*
- BornEvent - *the verb born or its tenses*
- DateEvent - *4 March 1951*
- LocationEvent - *Glasgow in Scotland*

A typical whole sentence pattern may be: *On 4 March 1951 in Glasgow in Scotland Kenny Matthieson Dalglish was born.* These patterns are taken from Wikipedia^{TM}. During the initial test runs, our system retrieved quite a good number of new triples. We manually went through a sample of the triples and found that there were quite a few with errors or false positives.

V. Evaluation

We analysed our results both using a known metric and by manually going through the new triples to check for validity. As a metric we used Precision & Recall (P&R) with the following criteria:

$$Precision = \frac{Relevant \quad triples \quad retrieved}{Number \quad of \quad triples \quad retrieved}$$

$$Recall = \frac{Relevant \quad triples \quad retrieved}{Number \quad of \quad triples \quad in \quad text}$$

This method of evaluation has been applied to the field of Information Retrieval such as extracting text summaries from news texts [17]. In IE, no evaluating algorithm is perfect due to contradictions in the texts and the nuances of the language [18].

Our initial evaluation of the results gave very low values of 0.21 and 0.13 for precision and recall respectively. In view of these results we had to determine what was causing them to be so low.

We encountered errors due to different font encoding (ANSI, UTF-8), to characters that make the application throw exceptions etc. Other errors in the data resulted from:

- Annotations spanning sentences
- Partial annotations not being discarded
- False positives especially in DateEvents
- Order of annotations when passing through the Annie pipeline
- False positives overall

In order to get better results we rectified the above error instigators by modifying one of our Jape grammars and one of our most important methods so that we matched a more specific pattern in the left hand side of the grammar to ensure that the sentences retrieved *really* contain the four pattern parts we needed. We also added a more specific RegEx to match dates instead of relying on ANNIE's default Date type.

A. Modifications made

One of the main issues was that the annotations were spanning sentences and so we had to introduce a {Sentence} condition in our grammar. This made sure that we could select whole sentences as annotations to check within their span for our four part-pattern. The right hand side of our modified Jape grammar works by:

1) Retrieve a whole sentence annotation
2) Obtain an iterator over the annotation
3) Try to match the various pattern parts within the sentence annotation
4) Set a Boolean variable to true if a part is found
5) If all four pattern parts are found and all Flags are TRUE
    - put the parts together in a new type called AllParts
    - add a new rule called GetAllPartsRule

Other modifications were made in the performGateProcessing() method that now uses sentence annotations and then the algorithm searches over them for the pattern parts. This made sure that only the patterns within that sentence are selected.

These modifications gave their fruit as the results and the P&R values increased four-fold and five-fold for Precision and Recall respectively. In the following section we will only list the values recorded from the results tables.

B. Results after modifications

For Test 1 we used the same criteria as the initial test. This test gave the following results:

Table I
MODIFICATION FOR **PERFORMGATEPROCESSING()** METHOD - SOURCE: AUTHOR

```
1.Get sentence annotation - put in sentence Set
2.Get sentence offsets - put in pattern Set
3.Retrieve the current annotation, i.e. the part of
    pattern needed
4.If GetAllParts is true
5.  Get features and offsets
6.    For (each type of pattern)
7.    Check if the type's offsets are within the
        GetAllParts offsets
8.    If (YES) retrieve the annotation with those
        offsets from the Set & Process accordingly
10.Remove consumed annotation from annotation set
```

Test 1

$$Precision = \frac{8+27+100+336}{8+27+100+389} = 0.89$$

$$Recall = \frac{8+27+100+336}{9+29+102+585} = 0.65$$

For subsequent tests, and to test the system's robustness, we used different seed files with information coming mainly from the IMDB website (http://www.imdb.com). The results of these tests gave a precision of 0.97, 0.61, 1, 0.7 and 0.54. The recall measure gave values of 0.97, 0.61, 0.83, 0.7 and 0.5. Taking all the above precision and recall results whilst summing for an average, we get the following:

$$Precision = \frac{4.71}{6} = 0.79$$

$$Recall = \frac{4.26}{6} = 0.71$$

These successful results were only possible with the code modifications we made after the initial runs of the application. This meant that when we used the same seed file during the development of the application and were getting good responses from the system we were failing to realise that the system was not as robust as it should have been. These shortcomings then manifested themselves after we used larger input files that gave many errors.

With these modifications the application retrieved less results, which was the only disadvantage we noticed. Apart from that we noticed that the results, although less, have more quality in that they are more correct. There are also less results that contain error or erroneous data and this can be verified from the Precision & Recall values we recorded. The quality of the triples we retrieved varied greatly although the quantity is less. We are of the opinion that less does not mean worse, we rather have less triples that are of very good quality rather than have large quantities whose quality is poor.

C. A note on Precision & Recall

According to [19], Precision & Recall results are supposed to be inversely proportional. In [20] it was found that precision and recall applied to data from news sources (data having some structure), places P&R values in the high 80s or low 90s (percentage-wise). In another study by [21], similar to ours, it was found that precision and recall are not always inversely proportional but may also produce values close to our results.

VI. Achievements & Limitations

The main aim of our research was to try and increase the chances of new triples being extracted from unstructured text or documents.

A. Achievements

In order to reach our set goals we produced an application that although having a simple design meets our goals' needs. The application makes use of embedded code from the GATE system and this permitted our application to generate the acceptable output we had. We also wrote two new JAPE grammars that helped us find our required four pattern parts and then use them to check whether these are contained in a given sentence that was also extracted from the same text. With the encouraging results we obtained after we modified our code, we have reached our objectives.

B. Limitations

During our research and application development we also encountered some difficulties that we did not have control upon but which made our development a little more interesting.

Our application currently operates with both the SPARQL and the Web search query code hard-coded within the application's methods. We intended for these to be dynamic and left to the user to decide which of the pattern parts to use in the query. This would have given the user a better usage experience.

In this application we are also not inserting on updating the dataset in the database due to certain factors such as the retrieved person names' ambiguity. We are accepting each new triple at face value as long as they are valid and correctly built. For example, *Steven George Gerrard* and *Steven "Stevie" Gerrard* may be found as being two different persons when in actual fact they are not. To overcome this the application would need other grammars that would have been outside of the scope of this research.

We are also not checking for duplicate findings so that in our result files we can find more that one entry with the same name. This happens because search engines may return similar result snippets that we are appending in one file and subsequently using these files as input for subsequent runs and tests.

VII. Future Work

On the whole, the application produced good results. Because of this, there are two areas where it needs improvements. First and foremost, we need to test the application on other patterns. At the moment, the patterns used were limited to identifying birth date and locations whilst associating them to a person. This is very useful information howerver there are other areas which one could explore such as working relations, educational relations, family relations, etc. In particular, we should go a step further and explore ways of generating these initial patterns automatically thus ensuring that the system is fully automated. This can be done by delving further into the creation of personal ontologies automatically.

Secondly, we need to test the system on a larger dataset. In particular it would be ideal to test it on the whole DBPedia corpus. Obviously, this is not a trivial thing to do especially since we require massive investment in computing resources. However the recent proliferation of cloud computing means that these massive requirements are finally within reach. Thus, our next project will involve the utilisation of the could to generate even more Linked Data.

VIII. Conclusion

Our research tried to exploit information extraction in order to generate linked data automatically thus realise Tim Berners-Lee's vision of the future web. A web made up of human readable documents together with documents that permit machines to also *understand* the documents they display and process. In doing so, the machine can be put in a better position to help its human user in doing more in a shorter span of time and in attaining more by the direct help of the machine itself.

Our system managed to extract patterns from a human readable text and use it to find similar patterns elsewhere by searching the WWW. This was achieved by using bootstrap techniques. The new data generated is then stored in RDF format and reused in other searches.

The evaluation conducted also produced extremely good results having an overall precision of 79% and a recall of 71%, thus encouraging us to look further into similar approaches. In conclusion, we demonstrated that semantic meaning can be extracted from natural text and that Linked Data can be generated with little effort. Our data can then be easily added to the growing datasets of the Linked Open Data cloud thus bring the Semantic Web a step closer to reality.

REFERENCES

[1] M. Sweeney, "Internet overtakes television to become biggest advertising sector in the UK," *The Guardian - newspaper*, 2009, last accessed 6th September, 2011.

[2] PricewaterhouseCoopers, "Internet Advertising Revenue Report," Internet Advertising Bureau, Tech. Rep., 2010.

[3] T. BernersLee, J. Hendler, and O. Lassila, "The semantic web - a new form of web content that is meaningful to computers will unleash a revolution of new possibilities," Scientific American Magazine - May 2001, 2001.

[4] T. Berners-Lee, "Putting government data online," 2009, personal view only.

[5] T. Berners-Lee, J. Hollenbach, K. Lu, J. Presbrey, and M. Schraefel, "Tabulator redux: Browsing and writing linked data," 2008.

[6] T. Berners-Lee, "Linked data - design issues," 2006, author's personal view.

[7] R. Cyganiak and A. Jentzsch, "Linking open data cloud," Richard Cyganiak and Anja Jentzsch, 2011, last accessed 3rd September, 2011.

[8] G. Kobilarov, T. Scott, Y. Raimond, S. Oliver, C. Sizemore, M. Smethurst, C. Bizer, and R. Lee, "Media Meets Semantic Web - How the BBC uses DBpedia and Linked data to Make Connections." in *ESWC*, ser. Lecture Notes in Computer Science, L. Aroyo, P. Traverso, F. Ciravegna, P. Cimiano, T. Heath, E. Hyvonen, R. Mizoguchi, E. Oren, M. Sabou, and E. P. B. Simperl, Eds., vol. 5554. Springer, 2009, pp. 723–737.

[9] D. Lange, C. Böhm, and F. Naumann, "Extracting structured information from wikipedia articles to populate infoboxes," in *Proceedings of the 19th ACM international CONFERENCE on Information and knowledge management*, ser. CIKM '10. New York, NY, USA: ACM, 2010, pp. 1661–1664.

[10] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "DBpedia: A nucleus for a web of open data," in *Proceedings of the 6th International Semantic Web Conference (ISWC)*, ser. Lecture Notes in Computer Science, vol. 4825. Springer, 2008, pp. 722–735.

[11] P. Jackson and I. Moulinier, *Natural Language Processing for Online Applications: Text Retrieval, Extraction, and Categorization (Natural Language Processing, 5)*. John Benjamins Pub Co, 2002.

[12] A. M. Turing, "Computing Machinery and Intelligence," *Mind*, vol. LIX, pp. 433–460, 1950.

[13] J. Weizenbaum, "Eliza&mdash;a computer program for the study of natural language communication between man and machine," *Commun. ACM*, vol. 9, pp. 36–45, January 1966.

[14] T. Winograd, "Procedures as a representation for data in a computer program for understanding natural language," *Cognitive Psychology*, vol. 3, no. 1, pp. 1–191, 1972.

[15] S. Sarawagi, "Information extraction," *Found. Trends databases*, vol. 1, pp. 261–377, March 2008.

[16] H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, N. Aswani, I. Roberts, G. Gorrell, A. Funk, A. Roberts, D. Damljanovic, T. Heitz, M. A. Greenwood, H. Saggion, J. Petrak, Y. Li, and W. Peters, *Text Processing with GATE (Version 6)*, 2011.

[17] J. Goldstein, M. Kantrowitz, V. Mittal, and J. Carbonell, "Summarizing text documents: sentence selection and evaluation metrics," in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '99. New York, NY, USA: ACM, 1999, pp. 121–128.

[18] C. Welty and J. Murdock, "Towards knowledge acquisition from information extraction," in *The Semantic Web - ISWC 2006*, ser. Lecture Notes in Computer Science, I. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. Aroyo, Eds. Springer Berlin / Heidelberg, 2006, vol. 4273, pp. 709–722.

[19] R. Baeza-Yates and G. H. Gonnet, *Modern Information Retrieval*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., 1999.

[20] N. Indurkhya and F. J. Demerau, *Handbook of Natural Language Processing, Second Edition (Chapman & Hall/CRC Machine Learning & Pattern Recognition)*. Chapman and Hall/CRC, 2010.

[21] A. Dingli, F. Ciravegna, and Y. Wilks, "Automatic semantic annotation using unsupervised information extraction and integration," in *Proceedings of the Workshop on Semantic Annotation and Knowledge Markup, SEMANNOT2003, K-CAP 2003 Workshop, at Sanibel, 26 October 2003*, S. Handschuh, M.-R. Koivunen, R. Dieng, and S. Staab, Eds., 2003.