RealText$_{lex}$: A Lexicalization Framework for Linked Open Data

Rivindu Perera, Parma Nand, and Gisela Klette

School of Computer and Mathematical Sciences,
Auckland University of Technology, New Zealand
{rperera,pnand,gklette}@aut.ac.nz

Abstract. Linked Open Data (LOD) is growing rapidly as a source of structured knowledge used in a variety of text processing applications. However, the applications using the LOD need to be able to mediate between the front end user interfaces and LOD. This often requires a natural language interpretation of this structured, linked data. We demonstrate a middle-tier framework that can generate patterns which can be used to transform LOD triples back into natural text. The framework utilizes preprocessed free text to extract a wide range of relations which are then aligned with triples to identify possible lexicalization patterns. These lexicalization patterns can then be used to transform a given triple into natural language sentence.

1 Introduction

The Linked Open Data encodes an enormous amount of knowledge in a structured form as triples making it machine readable. However, consumption of this rich resource in applications is often hindered by the inability to represent this encoded knowledge as a natural language [1]. This has created the need for a middle-tier framework which enables us to transform the Linked Data triples into natural text.

The purpose of RealText$_{lex}$ framework[1] is to transform Linked Data triples into natural language sentences which is often referred to as the process of lexicalization. The framework is part of our RealText language generation system as a support component, however it can also be used as a stand-alone framework. The main goal of the framework is to derive patterns that can be used to transform the triples into natural text. However, we also draw attention to the features of Linked Data which spawns a range of linguistic patterns. The approach underlying the RealText$_{lex}$ is presented in [2], which provides details of the experiments and the analysis of the results. All features presented herein will be part of the demonstration.

[1] A video demonstration is available at https://vimeo.com/user41119759/realtextlex

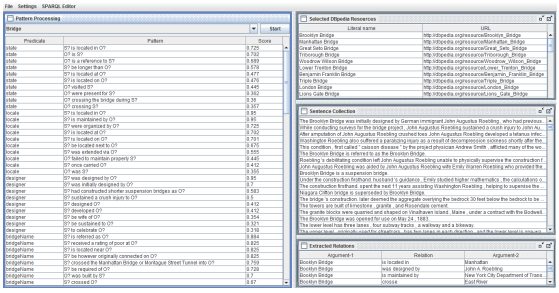

Fig. 1. RealText desktop application. The patterns extracted are shown in the left grid window. The three stacked windows in right show the selected DBpedia resources, candidate sentences, and extracted relations.

2 Demonstration

The goal of the demonstration will be to show the whole of the RealText$_{\text{lex}}$ workflow from the extraction and preprocessing of unstructured text to the generation of lexicalization patterns. The demonstration will utilize a Java client application built on top of the RealText$_{\text{lex}}$ framework. In essence, the demonstration will focus on how the lexicalization patterns can be extracted from the unstructured text and the techniques of utilizing extracted patterns to transform a given Linked Data triple into natural text.

2.1 Datasets

For the purpose of this demonstration we focus on triples belonging to five randomly selected DBpedia ontology classes; Bridge, Actor, Publisher, River, and Radio Host. These five classes offer 132 unique predicates which can be lexicalized. A detailed description on the datasets used to evaluate the framework can be found in [2].

2.2 Workflow

The workflow underlying the RealText$_{\text{lex}}$ comprises of four main steps: (1) Candidate sentence extraction, (2) Relation extraction and alignment, (3) Ensemble pattern processing and combination, and (4) Pattern enrichment and persistence.

Candidate sentence extraction The input to the candidate sentence extraction module is an ontology class represented in DBpedia (e.g., Bridge). This module attempts to select a random set of DBpedia RDF files (e.g., Brooklyn_Bridge.rdf) which belong to the given ontology class and extracts triples from the selected Resource Description Framework (RDF) files.

In the next phase, the module extract text related to the selected DBpedia RDF file (e.g., Wikipedia page for Brooklyn Bridge). The main source of free

text extraction are the Wikipedia pages for selected DBpedia RDF file. However, Wikipedia on its own is not sufficient to create a rich enough natural language representation of RDF triples. In order to enrich the possible patterns, a web search module (based on Bing API[2]) was integrated to extract more text snippets related to the RDF file from websites. The extracted text was cleaned using the shallow text feature based boilerplate removal algorithm[3] and the co-references were resolved using the Stanford CoreNLP.

The resulting sentence collection from above step was analysed to determine the candidate sentences that have RDF triple elements corresponding to subject, predicate, and object. The candidate sentences were then assigned to triples with a score proportional level of presence of the individual triple elements.

Relation extraction and alignment In the next step we extracted the relations ($arg_1 \leftrightarrow$ rel $\leftrightarrow$ arg_2) from the generated candidate sentences from the previous step. For this we employed the self-supervised Open Information Extraction (Open IE) module[4] which enables us to identify a large number of relations and to associate them with triples. Our hypothesis is that a relation can be aligned with a triple structure to identify potential lexicalization of the triple under consideration. We aligned each relation with the triple and assigned a score based on the level of alignment. The aligned relations were then processed to extract the patterns which were used to lexicalize the triple as natural text.

Ensemble pattern processing and combination In this phase, we first transformed the aligned relations to patterns by substituting subject and objects literal values with generic expressions (e.g., $\langle arg_1$: Brooklyn Bridge, rel: is designed by, arg_2: John Roebling$\rangle \Rightarrow$ *s?* is designed by *o?*). The success of Open IE based pattern extraction is based on the availability of the textual representation of the required triple. Therefore, we consider alternatives to generating patterns which work with the main module. For this we utilized the verb frames from lexical semantic resources, mainly from WordNet and VerbNet. The pattern processing module generates patterns that can be extracted from relations followed by the attempt to lookup the predicate in the verb frame database[5]. If the module is able to find a pattern of NP$\leftrightarrow$VP$\leftrightarrow$NP structure for the given verb, then a predetermined pattern of [subject]$\leftrightarrow$VP$\leftrightarrow$[object] is created. A predetermined set of properties of entities (e.g., eye colour, hair colour) are lexicalized with a pre-processed set of patterns.

Pattern enrichment and persistence Some features of triples can result in patterns that cannot be generalized to other triples with the same predicate. We

[2] `http://www.bing.com/toolbox/bingsearchapi`
[3] `https://code.google.com/p/boilerpipe/`
[4] `http://knowitall.github.io/ollie/`
[5] An embedded database which contains verb frames from WordNet and VerbNet.

have identified three such features in our investigations; grammatical gender, fine ontology class, and object multiplicity. A detailed explanation of how these features affect patterns to be in a different form is given in [2]. The generated patterns are stored in a SQLite database and version 1.3 of this database can be found on the project web site[6].

3 Related Work

Recent frameworks like Lemon[3] and LOD-DEF[4] have also studied the lexicalization problem with a particular focus on Linked Data. However, these systems lack three main concepts which were presented in this paper; the use of rich textual sources for pattern extraction (including boilerplate removal and coreference resolution), the cohesive pattern generation, and the investigation of encoded features. For instance, the concept recently proposed in Lemon is based on dependency paths to find the pattern which will eventually raise the need for a large syntactic rule set. Furthermore, we have brought up the need for identifying features (for more information see [2]) before coming up with frameworks to generate patterns.

4 Conclusion and future work

The approach for LOD lexicalization presented in this paper offers new insights into generating lexicalization patterns for Linked Open Data cloud by applying Open IE on automatically collected and processed free text. The lexicalization module described here is part of a parent Natural Language Generation project [5,6] currently underway at Auckland University of Technology. In future, we expect to expand our framework by analysing the missing features that can affect the lexicalization patterns and to improve the text extraction and processing module to support extracting a large number of accurate lexicalization patterns.

References

1. Perera, R.: Scholar: Cognitive Computing Approach for Question Answering. Honours thesis, University of Westminster (2012)
2. Perera, R., Nand, P.: A multi-strategy approach for lexicalizing linked open data. In: CICLing-2015. (2015) 348–363
3. Walter, S., Unger, C., Cimiano, P.: A corpus-based approach for the induction of ontology lexica. In: NLDB-2013. (2013) 102–113
4. Duma, D., Klein, E.: Generating natural language from linked data: Unsupervised template extraction. In: IWCS-2013. (2013)
5. Perera, R., Nand, P.: Real text-cs - corpus based domain independent content selection model. In: ICTAI-2014. (2014) 599–606
6. Perera, R., Nand, P.: The role of linked data in content selection. In: PRICAI-2014. (2014) 573–586

[6] `http://rivinduperera.com/information/realtextlex.html`