Speeding up multilingual grammar development by exploiting linked data to generate pre-terminal rules

Sebastian Walter, Christina Unger, and Philipp Cimiano

Semantic Computing Group, CITEC, Bielefeld University

Abstract. The development of grammars, e.g. for spoken dialog systems, is a time- and effort-intensive process. Especially the crafting of rules that list all relevant instances of a non-terminal, e.g. Greek cities or Automobile companies, possibly in multiple languages, is costly. In order to automatize and speed up the generation of multilingual terminal lists, we present a tool that uses linked data sources such as DBpedia in order to retrieve all entities that satisfy a relevant semantic restriction. We briefly describe the architecture of the system and explain how it can be used by means of an online web service.

1 Introduction

The development of grammars, e.g. for spoken dialog systems (SDS), is a costly process requiring large manual investments [2]. One aspect of SDS grammar development that is particularly costly is the process of developing pre-terminal rules, i.e. rules that expand a non-terminal into a number of named entities. For example, when developing a dialog system that provides access to the bus schedule connecting Greek cities, one needs a list of names of Greek cities. Similarly, when developing a dialog system that is able to perform conversions between different currencies, one needs a list of the relevant currencies that exist worldwide. In such cases we would like to create pre-terminal rules like the following ones:

```
1 GREEK_CITY = Athens | Thessaloniki | ...
2 CURRENCY   = Euro | Dollar | Yen | ...
```

The acquisition of lists of such entities is costly, and if the grammar needs to be developed for different languages, this problem is exacerbated.

On the other hand, a massive amount of structured and interlinked knowledge is currently emerging in the form of the Linked Open Data cloud[1].

In this paper we present an approach that supports the acquisition of lists of named entities in order to speed up the process of creating pre-terminal rules. The approach exploits DBpedia as the central hub of the Linked Open Data cloud. Given one or more examples, it retrieves the classes that those examples

[1] http://lod-cloud.net/

share, as well as properties that they have in common. The user can select those classes and properties that are relevant, which are then used to retrieve all available entities and returns them in form of a grammar rule in ABNF format that can be integrated into the grammar project in question.

The paper is structured as follows. In Section 2 we provide some information about DBpedia. Section 3 then briefly describes our approach. We conclude in Section 4. The source code can be found at `https://github.com/swalter2/TerminalEnhancement`. To run the system a DBpedia SPARQL endpoint is needed, such as the official DBpedia SPARQL endpoint[2] with DBpedia 3.9. A link to a live demo of our system can be found in the GitHub repository.

2 Linked Open Data and DBpedia

The Linked Open Data cloud consists of a large amount of interlinked RDF[3] (Resource Description Framework) datasets, including knowledge bases such as DBpedia[4] and YAGO[5]. It has been growing steadily in recent years, now comprising more than 30 billion RDF triples[6].

The central hub of the Linked Open Data cloud is DBpedia [1], a cross-domain knowledge base that was extracted from Wikipedia infoboxes. The English version of DBpedia currently comprises around four million entities, most of them organized in a consistent ontology. This ontology includes labels in a range of languages. In addition, there are more than 100 localized versions of DBpedia available. YAGO is another knowledge base extracted from Wikipedia, which in contrast to DBpedia also includes information from WordNet[7] in order to categorize entities.

Such structured knowledge bases become more and more popular for various applications. However, to our knowledge, such structured knowledge bases have not been exploited to speed up spoken dialogue grammar development. In order to show the potential of the structured data available on the web for rapid grammar development, we present a tool that uses DBpedia in order to automatically generate terminal rules, given some semantic restrictions provided by a grammar developer.

3 Demonstration and system description

This section briefly describes the tool that will be demonstrated at the conference. It uses one or more resources as input and returns a terminal grammar in ABNF format as output. Figure 1 provides an overview of the architecture.

[2] `http://dbpedia.org/sparql`
[3] `http://www.w3.org/TR/rdf-primer/`
[4] `http://dbpedia.org`
[5] `http://www.mpi-inf.mpg.de/yago-naga/yago/`
[6] `http://lod-cloud.net/state/`
[7] `http://wordnet.princeton.edu/`

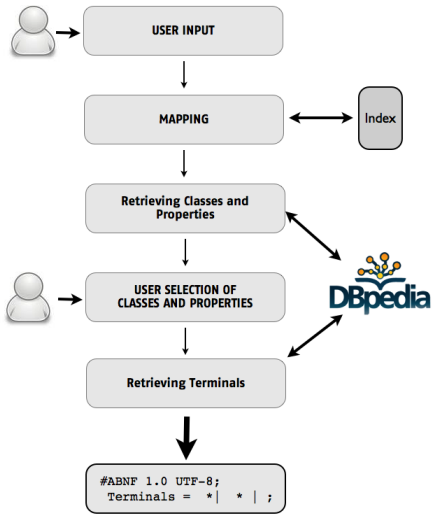

Fig. 1. System overview

First, the user has to enter the name of at least one (but optimally two or more) resources. In the following, we will use the input *Bruce Lee* and *Jackie Chan* as example. Currently this input is matched directly with DBpedia resource labels, but we will shortly add an index lookup that includes anchor texts from Wikipedia and thus allows also for indirect matches, e.g. using as input *Chan* or *Jacky Chan*. Given the input resources, the system retrieves all DBpedia and YAGO classes they have in common. In addition, all classes that belong to the DBpedia ontology namespace are sorted hierarchically, using the property `rdfs:subClassOf`. The most general class is presented at the top (in the case of *Bruce Lee* and *Jackie Chan* the class `Person`), the most special class is presented at the bottom (in this case the class `Actor`). In addition, the number of entities of the corresponding class are indicated in brackets after the class name. As the class names are not always very comprehensible, e.g. `Site108651247`, five example resources are provided for each class. The YAGO classes are not connected through `rdfs:subClassOf`, such that they are sorted by the number of elements they have. Furthermore, all properties for which the input resources share the object are returned. In the case of *Bruce Lee* and *Jackie Chan*, for instance, this would be the property `country` with object `China`.

The retrieval step works for an arbitrary number of (non-zero) input resources. This means that if only one resource is given, all types of that resource are retrieved, together with all property-object pairs of the triples it occurs in as subject.

Next, the user has different options to proceed:

1. choosing one or more relevant property-object pairs
2. choosing one or more classes
3. choosing one or more classes and also one or more relevant property-object pairs

If more than one class is selected, the user can furthermore choose if the semantic restriction should be created by connecting all classes using AND or OR semantics. The selected classes and properties are then used to retrieve all resources that match those constraints. Finally, the labels of those resources,

possibly in different languages, are returned as the target terminals. Depending on the chosen constraint(s), a different number of resources (terminals) are returned. For our example of *Jackie Chan* and *Bruce Lee*, option one amounts to choosing `country` with object `China`, returning around 6,250 terminals. Option two would amount to selecting one or both of the classes `Actor` and `Person`. When connecting both classes by a logical AND, 2,669 terminals are returned; connecting them with a logical OR returns around 50,000 terminals. Option three returns 903 terminals when choosing both classes and the property-object pair, i.e. asking for all persons who are actors from China.

Besides the default language English, the user can currently choose one additional language, such as Spanish, German, Russian or Chinese. In case a terminal has only an English label, but none for the selected second language, only the English label is returned. In the demo we restricted the language choice for performance reasons, but in principle the system can return terminals in arbitrarily many different languages, as long as they are supported by DBpedia.

Finally, all returned terminals are displayed in the browser and for each language an ABNF grammar file is generated, which can be downloaded. We chose ABNF[8], as this is one of the most common SDS grammar formats, but the tool could be easily adapted to include other grammar formats as well, e.g. GRXML[8].

4 Conclusion

In this paper we presented a first approach of an easy-to-use tool that has the potential to significantly speed up the development of multilingual grammars by exploiting linked data for the generation of pre-terminal rules. As an evaluation of the tool, we plan to compare the amount of time needed by a grammar expert to create a terminal grammar by hand compared to the amount of time need when being supported by our tool. This will provide us a detailed picture of the efficiency of our approach.

Acknowledgment This work has been funded by the European Union's Seventh Framework Programme (FP7-ICT-2011-SME-DCL) under grant agreement number 296170 (PortDial).

References

1. Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. DBpedia – a crystallization point for the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154–165, 2009.
2. Giuseppe Riccardi, Philipp Cimiano, Alexandros Potamianos, and Christina Unger. Up from limited dialog systems! In *NAACL-HLT Workshop on Future Directions and Needs in the Spoken Dialog Community: Tools and Data*, pages 1–2. Association for Computational Linguistics, 2012.

[8] `http://www.w3.org/TR/speech-grammar/`