FORGe at WebNLG 2017

Simon Mille and Stamatia Dasiopoulou
Universitat Pompeu Fabra, Roc Boronat 138, 08018 Barcelona, Spain
firstname.lastname@upf.edu

Abstract

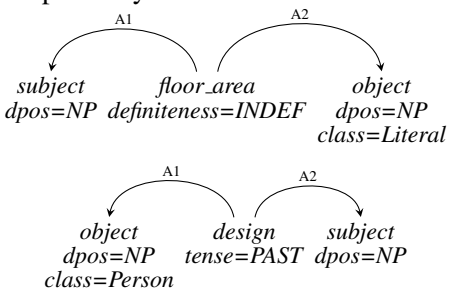
This paper describes the FORGe generator at WebNLG. The input DBpedia triples are mapped onto sentences by applying a series of rule-based graph-transducers and aggregation grammars to template predicate-argument structures associated to each property.

1 Introduction

For the WebNLG challenge, the FORGe generator, which was originally designed to take linguistic predicate-argument (PredArg) structures as input (Mille et al., 2017), has been further developed in order to support RDF triples as input. The provided data was XML files that contained between one and seven DBpedia triples of one particular category. Half of the evaluation data consisted of categories found in the training data, and the other half consisted of new categories with mostly unseen properties. The generation process involves the following steps: (i) mapping of DBpedia properties onto PredArg templates; (ii) template population; (iii) sentence planning; (iv) linguistic generation.

2 Mapping of DBpedia properties to PredArg templates

Previous to receiving the evaluation dataset, we defined 218 predicate-argument templates taking into account the property as well as the type of the subject and object values.[1] Thus, we associated each of the 297 distinct DBpedia properties found in the original triples of the training data to one of these templates. Parts of speech (e.g., NP (proper noun)), grammatical features (e.g., verbal tense or nominal definiteness), or information from DPpedia (e.g., classes), for instance, can be specified in the template.[2] The following figure shows sample PredArg templates associated to the properties *author* and *architect* respectively.



We then extended these templates in order to cover the unknown properties included in the evaluation data. We used a total of 318 templates for generating the whole evaluation data.

3 Population of the templates

Using the aforementioned mappings between DBpedia properties and predicative templates, each input triple is transformed into a respective PredArg structure. This involves two main steps. First, the cleaning of the object, including the extraction of value/unit information from datatype fillers and distinct values from list-like fillers. Second the assignment of pertinent subject/object class labels; these are geared to the subsequent linguistic generation

[1] Looking at subject and object types was needed as some properties denoted more than one meanings and corresponded to different templates.

[2] Unspecified values are assigned as needed later in the generation process.

steps and currently include Person, Location, Time (further distinguishing between date, year, month) and Literal (i.e. datatype values). During this step, cardinality and plurality information labels are also assigned. Last, in the case of inputs of multiple triples, these are ordered based on the number of appearances of their subjects and on whether a subject of a triple serves also as an object in another, as a preliminary step for the subsequent aggregation.

4 Aggregation of PredArg structures

Feeding the PredArg structures as such to the FORGe generator would render each triple as an independent sentence. In order to group triples into complex sentences , we implemented a new graph-transduction module for FORGe that performs aggregation in two steps.

First, we look for shared pairs of predicate and subject argument in the populated templates: if the object arguments have the same relation with their respective predicates, they will be coordinated (e.g., *Jazz$_S$ influenced$_P$ funk$_{O1}$ and afrobeat$_{O2}$.*); if the relations are different, the objects become siblings under the first occurrence of the predicate (e.g. *Alan Bean$_S$ was born$_P$ [in Wheeler (Texas)]$_{O1}$ [on March 15]$_{O2}$.*); the duplicated nodes are removed.

Second, we check if an argument of a predicate appears further down in the ordered list of PredArg structures. If so, the PredArg structures are merged by fusing the common argument; during linguistic generation, this results in the introduction of post-nomimal modifiers such as relative and participial clauses or appositions (e.g. *[The home town]$_{P1}$ of John Madin$_S$, who designed$_{P2}$ [103 Colmore Row]$_{O2}$, is Birmingham$_{O1}$.*). In order to avoid the formation of heavy nominal groups, we allow at most one aggregation by argument. Referring expressions are introduced during linguistic generation.

5 Linguistic generation

The next and last step is the rendering of the aggregated PredArg structures into sentences. For this, we use the core FORGe grammars (Mille et al., 2017), but replace statistical linearization by a rule-based linearization component, in order to have more control over the output quality. This part of the system follows the theoretical model of the Meaning-Text Theory (Mel'čuk, 1988), and performs the following actions: (i) syntacticization of predicate-argument graphs; (ii) introduction of function words; (iii) linearization and retrieval of surface forms.

First, a deep-syntactic structure is generated: missing parts of speech are assigned, the syntactic root of the sentence is chosen, and from there a syntactic tree over content words is built node by node. Then, idiosyncratic words (prepositions, auxiliaries, determiners, etc.) are introduced and fine-grained (surface-)syntactic labels are established. For this, we use a valency (subcategorization) lexicon, but due to time constraints, we sometimes indicated which functional prepositions to use as a feature in the template in order to overrule an erroneous or missing entry. Personal and relative pronouns are introduced using the *class* feature that allows for distinguishing between human and non-human antecedents. Finally, morpho-syntactic agreements are resolved, the syntactic tree is linearized (ordering of (i) governor/dependent and (ii) dependents with each other) and the surface forms are retrieved.

6 Future work

The aggregation and generation of referring expressions modules will be refined; for instance, now we only use string matching to generate pronouns, and do not control the possible ambiguities about the antecedent. We will also explore the further use of the information and semantics provided by the DBpedia ontology, for instance, for having access to gender information and different ways of referring to the same named entity.

Acknowledgments

This work has been partially funded by the European Commission under the contracts H2020-645012-RIA, H2020-700024-RIA, and H2020-700475-RIA.

References

Igor Mel'čuk. 1988. *Dependency Syntax: Theory and Practice*. SUNY Press, Albany.

Simon Mille, Roberto Carlini, Alicia Burga, and Leo Wanner. 2017. Forge at semeval-2017 task 9: Deep sentence generation based on a sequence of graph transducers. In *Proceedings of SemEval-2017*, pages 917–920, Vancouver, Canada, August. Association for Computational Linguistics.