Multi-Language Surface Realisation as REST API based NLG Microservice

Andreas Madsack, Johanna Heininger, Nyamsuren Davaasambuu, Vitaliia Voronik, Michael Käufl and **Robert Weißgraeber**

AX Semantics, Stuttgart, Germany
{firstname.lastname}@ax-semantics.com

Abstract

We present a readily available API that solves the morphology component for surface realizers in 10 languages (e.g., English, German and Finnish) for any topic and is available as REST API. This can be used to add morphology to any kind of NLG application (e.g., a multi-language chatbot), without requiring computational linguistic knowledge by the integrator.

1 Introduction

The proliferation of chatbots changed the way we see user interfaces. Alexa, Siri or Cortana want to draw us into a conversation, causing two challenges from a "natural" language perspective: grammar and context. Services like RASA or Google's Dialogflow, only provide the use of templates for the responses. In languages where the grammatical agreement is mostly represented by a variation for singular or plural this is fairly easy. However, for a language like German, French, or even Russian you have to write a large set of templates – or simplify your answers, consequently restricting your creativity and eliminating the naturalness of the responses.

Having implemented a few successful projects building chatbots using our NLG system (Weißgraeber and Madsack, 2017), we saw the need to open up some underlying parts of our system to enable deeply integrated and componentized approaches. Using existing complete NLG systems for projects like chatbots works, but demands learning to use a system designed not to generate dialogues but complete articles. Instead of using our NLG system you may want a simpler way for story building, and possible integrations with the chatbot-serving app and data, for adding context from data signals while rendering your text response.

Whatever you are using to generate context, response interference and response context – for example from results given based on the parse of the Natural Language Understanding system you are using – your response has to be enriched with grammatical agreement.

2 Example: Dialogue interface for an e-commerce system

As an example, we will use a fictional order status update from an e-commerce system sent to the customer. The system should respond to the customer by using a fully grammatical sentence – and not only a list of products – in the customer's language. These messages can be personalized and used with mixed (unknown) languages and products where a templating approach can not guarantee to be grammatically accurate.

In English a potential response message could be: "We confirm your order of two new black TVs and an Amazon Echo".

3 Solution

In this paper we describe an approach that offers an intermediate solution for this problem, that is neither about writing simple templates, nor using a complete NLG system. Instead it uses a "Grammar API" microservice for the morphology components of the surface realiser and otherwise uses openly available software tools in the application chain.

This API offers three endpoints to allow granular access to morphological components.

The first one is *verb*. Given *tense, number, person*, and *gender* the inflected verb is returned. For example in Finnish providing *lemma=sanoa, tense=present, number=s*, and *person=1st* returns *sanon* (*to say* → *I say*).

Proceedings of The 11th International Natural Language Generation Conference, pages 480–481,
Tilburg, The Netherlands, November 5-8, 2018.

language	lemma	parameters	result
en-us	TV	`adjs=[new, black] prep=of conj=' ' num=2 num_type=cardinal`	of two new black TVs
nl-nl	televisie	`adjs=[nieuw, zwart] prep=van num=2 num_type=cardinal`	van twee nieuwe, zwarte televisies
fi-fi	televisio	`adjs=[uusi, musta] case=ela num=2 num_type=cardinal`	kahdesta uudesta, mustasta televisiosta
ru-ru	телевизор	`adjs=[новый,чёрный] num=2 case=gen num_type=cardinal`	двух новых, чёрных телевизоров

Table 1: Examples for API parameters with resulting nounphrases

The second api endpoint is *quantifier*. The numeral_type parameter sets ordinal or cardinal and also the value of the numeral needs to be set: e.g., in Dutch providing *num_type=ordinal, num=3*, and *gender=f* returns *derde* (*third*).

The third and most interesting endpoint is *nounphrase*. A nounphrase consists of a noun and optionally also of one or more adjectives, a determiner, a quantifier, and/or a preposition. The need for case, gender, or animacy depends on the language. See table 1 for some examples.

3.1 API backend

The inflection is based on grammatical algorithms that run through a decision chain for each request, where all grammatical features of the language are implemented.

Corner-cases are covered by using lexicon entries. Grammatical features are implemented globally in a general way, and then added in each language by its individual configuration.

Languages covered by the first release of the Grammar API include English, German, French, Dutch, Spanish, Portuguese, Italian, Czech, Russian, and Finnish.

4 Related Work

Especially in academic works, the reference system to embed a surface realizer into one's own projects is *SimpleNLG* (Gatt and Reiter, 2009), mostly because it is the only openly accessible component.

Having to add a local library into your system does not follow modern software architecture methodologies, where cloud based services are readily available to cover all tasks of your toolchain and only need to be plugged in together with glue code, e.g. Dialogflow.

5 Availability

The Grammar API is available as a commercial web service for industrial applications, with English offered for free. Academic institutions and researchers however can get access to the API for all languages at no cost. For more information see `https://301.ax/grammar-api/`.

6 Conclusion and Future Work

By providing access to the grammar subcomponents from our integrated NLG systems, emerging use cases where NLG is integrated into other systems can be supported, allowing to build NLG-enabled applications faster than ever by lowering the barrier-to-entry on computational linguistics know how for software developers.

In our current work we concentrate on two main aspects that will be released into the API:

(1) Backend optimisations inside the algorithms, replacing some of the rule-based methods with statistics-based components.

(2) Providing expanded API endpoints, e.g. making it possible to render complete parts of a sentence from POS-Tag information.

Additionally, our microservice can be used as a surface realizer for NLG tasks in other applications.

References

Albert Gatt and Ehud Reiter. 2009. SimpleNLG: A realisation engine for practical applications. In *Proceedings of the 12th European Workshop on Natural Language Generation (ENLG 2009)*, pages 90–93, Athens, Greece. Association for Computational Linguistics.

Robert Weißgraeber and Andreas Madsack. 2017. A working, non-trivial, topically indifferent nlg system for 17 languages. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 156–157. Association for Computational Linguistics.