JSrealB: A bilingual text realizer for web programming

Paul Molins, Guy Lapalme
RALI
Informatique et recherche opérationnelle
Université de Montréal
CP 6128, Succ. Centre-Ville
Montréal, Québec, Canada H3C 3J7
{molinspa,lapalme}@iro.umontreal.ca

Abstract

JSrealB is an English and French text realizer written in JavaScript to ease its integration in web applications. The realization engine is mainly rule-based. Table driven rules are defined for inflection and algorithmic propagation rules, for agreements. It allows its user to build a variety of French and English expressions and sentences from a single specification to produce dynamic output depending on the content of a web page.

Natural language generation can automate a significant part of textual production, only requiring a human to supply some important aspects and thus saving considerable time for producing consistent grammatically correct output. In recent years, tools such as SimpleNLG (Gatt and Reiter, 2009) facilitated text realization by a programmer provided they program their application in Java. This system was then extended with SimpleNLG-EnFr (Vaudry and Lapalme, 2013), a English-French version of SimpleNLG.

Another approach to text realization is JSreal (Daoust and Lapalme, 2014), a French Web realizer written in JavaScript. This paper describes an attempt at combining the ideas of SimpleNLG-EnFr and JSreal to produce a bilingual realizer for French and English from a single specification. JSrealB generates well-formed expressions and sentences. It can be used standalone for linguistic demonstrations or be integrated into complex text generation projects. But like JSreal, it is aimed at web developers, from taking care of morphology, declension and conjugation to creating well-formed texts. A web programmer who wishes to use JSrealB to produce flexible English and/or French textual or HTML output only needs to add two lines in the page: one for importing program and one for calling JSrealB loader to load the resources (i.e. lexicon and rules).

The principles underlying JSrealB are similar to those of SimpleNLG: programming language instructions create data structures corresponding to the constituents of the sentence to be produced. Once the data structure (a tree) is built in memory, it is traversed to produce the list of tokens of the sentence. This data structure is built by function calls whose names are the same as the symbols usually used for classical syntax trees: for example, `N` to create a *noun* structure, `NP` for a *Noun Phrase*, `V` for a *Verb*, `D` for a *determiner*, `S` for a *Sentence* and so on. Features added to the structures using the dot notation can modify the values according to what is intended.

JSrealBsyntactic representation is patterned after classical constituent grammar notations. For example,

```
S(NP(D("a"),N("woman")).n("p"),
  VP("eat").t("ps"))
```

is the JSrealB specification for *The women ate*. Plural is indicated with feature `n("p")` where `n` indicates *number* and `p` *plural*. The verb is conjugated to past tense indicated by the feature tense `t` and value `ps`. Agreement between `NP` and `VP` is performed automatically.

French and English are languages whose structures are similar. Both languages use the same alphabet, they are both fusional languages sharing a similar conjugation system and their word order follows the same basic Subject-Verb-Object paradigm (Shoebottom, 1996). But structural differences do exist: e.g. the position of adjectives differ and rules for gender and number agreement differ for nouns and pronoun between these languages.

These differences must be taken into account at many levels. First, syntactic differences and agreements (i.e. features propagation), must be handled at the phrase or sentence level by algo-

rithms. For French complex rules, we followed "Le bon usage, grammaire française" (Grevisse, 1980). For English, we relied on various sources from the web.

JSrealB lexicons are based on the ones found in SimpleNLG-EnFr (Vaudry and Lapalme, 2013) These lexicons can be completed by the user to add domain-specific vocabularies. In lexicons, words have grammatical properties (e.g. category, gender, etc.). and a link to an inflection table. Tables are defined for nouns, adjective, verbs and pronouns in both English and French. These inflection rules are language specific and correspond to the information found in (Bescherelle, 2012) and (Delaunay and Laurent, 2013). These conjugation, declension or transformation tables are included with the lexicon in JSrealB, they are defined declaratively for each language and interpreted by a rule engine common to both languages.

Our goal was to develop an English and French text realizer with minimal specific adaptations to each language. We have promoted the systematic application of rules hoping it is possible to support other languages at limited cost.

Text realization uses a syntactic hierarchical tree representation that creates a sentence by combining phrases and terminals. The relations between these lexical units determine the propagation of features between words for determining proper agreements. For example, in

```
S(NP(D("le"),
     N("monsieur").n("p")),
  VP(V("avoir").t("i"),
     NP(D("un"),N("souris"))))
```

grammatical categories of words are already specified in the syntactic representation. Word order usually follows the left to right order of the terminals in the tree except in some coordinated sentences where position of coordinate must be determined.

The relations between non-terminals specified in the input determine the grammatical functions of each element, which are roughly similar between French and English. We can then compute the agreement between elements of the sentence in order to propagate appropriate features to the words according to the rules of the language.

Orthographic realization is performed during morphological realization. Sentence relays features, especially HTML tags, capitalization and full stop, to its children elements with the aim of formatting each phrase with proper elision.

JSrealB implements French and English grammatical categories: noun, pronoun, determiner, adjective, preposition, conjunction, and complement; the implemented phrases are: noun, verbal, adjectival, adverbial, prepositional, subordinate, and coordinate. The sentence combines all these phrases.

Supported inflections are conjugation for simple tenses, and declension in gender and number for every grammatical category. Noun phrase agrees in gender and number, while verbal phrase agrees with every type of subject (i.e. common or proper noun, or pronoun).

JSrealB currently realize sentences structured in the Subject-Verb-Object paradigm (e.g. *It will rain tomorrow.*), or noun phrases (e.g. *Heavy snowfalls this night!*).

But there is still much work in order to obtain a more complete coverage. For example negation is not yet handled: in French, negation is realized with two adverbs *ne* and *pas* (e.g. *il ne parle pas*), while in English there is only one: *not* (e.g. *he does not speak*). Moreover, the proper placement of the adverb is quite intricate.

There are also contractions (e.g. *can not* sometimes contracts in *cannot* in English) and elision (e.g. *it is* become *it's*). These rules require information that should be inserted in the lexicon and algorithmic mechanisms to be implemented at the phrase level. Finally, some properties are not yet considered (e.g. if word is countable or uncountable), which can cause non colloquial realization (e.g. `NP(D("a"),N("milk"))` produce *a milk* instead of *the milk*).

We will proceed to add new rules and types of sentences. Nevertheless, the core of the program is well developed and tested and various extension mechanisms have been designed so that we can quickly achieve a better coverage.

Availability

Examples of the use of JSrealB, and a web-based development environment are available at:

http://rali.iro.umontreal.ca/rali/?q=en/jsrealb-bilingual-text-realiser

The javascript code of the realizer, the lexicon and tables and a user manual (in French) are made available to the NLG community at:

https://github.com/rali-udem/JSrealB

References

Bescherelle, *Bescherelle La conjugaison pour tous*, Hatier, 2012.

N. Daoust and G. Lapalme, "JSreal: A Text Realizer for Web Programming", *Language Production, Cognition, and the Lexicon*, Springer, 2014, pp. 363-378.

B. Delaunay and N. Laurent, *Bescherelle La grammaire pour tous*, Hatier, France, 2013.

B. Garner, *Garner's Modern American Usage*, Oxford University Press, 2009.

A. Gatt and E. Reiter, "SimpleNLG: A realisation engine for practical applications", in *12th European Workshop on Natural Language Generation*, Athens, Greece, 2009, pp. 90-93.

M. Grevisse, *Le bon usage, grammaire française,* 11e édition, Duculot, Louvain-la-Neuve, Belgique, 1980.

P.-L. Vaudry and G. Lapalme, Adapting SimpleNLG for bilingual English - French realisation, *14th European Workshop on Natural Language Generation*, Sofia, Bulgaria, 2013, pp. 183-187.