Intelligent Word Order Handling for Natural Language Production

Natural Language Processing 2017-2018 (Assignment 3)
Prof. Dr. Katrien Beuls (katrien@ai.vub.ac.be)

Natural language production aims to formulate semantic representations into concrete utterances. When producing these utterances, the ordering of the words is of great importance. In some cases, the word order can change the meaning of the utterance completely, and is thus is an important part of the grammar (e.g. “the dog bites the man” vs. “the man bites the dog”). In other cases, the word order is reigned by more subtle conventions (e.g. “a nice blue car” vs. “a blue nice car”). Word order constraints of the first kind will often be provided by the grammar formalism (in our case, Fluid Construction Grammar), whereas word order constraints of the second kind will often need to be inferred from corpus statistics. The process of ordering words into an utterance based on this kind of constraints is referred to as *rendering* inside the FCG community.

Task description

For this assignment, you will implement a render function that takes into account both kinds of constraints. Concretely, the rendering function will need to be implemented as a search process. The words that need to be ordered, as well as the constraints imposed by the grammar, will be provided as shown below. The render function should be written in a modular way, so that additional constraints can easily be added.

- `(string the-1 "the")` `(string nice-1 "nice")` `(string large-1 "large")` `(string house-1 "house")` `(string burns-1 "burns")`
- `(precedes the-1 house-1)` : precedence relation
- `(meets house-1 burns-1)` : left-right adjacency relation
- `(first the-1)` : first place in the utterance
- …

The ordering of words that are not constrained by the hard constraints provided by the grammar, should be ordered according to n-gram probabilities. You can download bi-gram models from the free N-grams made available through the Corpus of Contemporary American English (COCA) (https://www.ngrams.info/download_coca.asp).

Your function should (at least) comply with the following criteria:

1. The render function takes a set of string and ordering predicates, and returns an utterance.
2. The utterance should contain all strings, and the constraints stated by the ordering predicates should

be satisfied.

3. The rendering function should be implemented as search process. It should be modular (easy to extend to new predicates) and well-documented.
4. The words which are not subject to ordering constraints should be ordered by their n-gram probabilities.

Additionally, you might want to implement the following functionality:

5. Warn the user if the string and ordering constraints that he entered lead to an utterance with a very high perplexity.

6. Add words into an utterance if they dramatically lower the perplexity (e.g. “The bird sits in tree” -> “sits in a tree”).

You will be evaluated on the quality and soundness of your algorithm, including its performance, accuracy, programming style and documentation. The program should preferrably be implemented in Common Lisp, which should be familiar to those that know Scheme. Alternatively, you can use Python.

Form

Write a short paper (max. 10 pages) that starts with a clear problem statement (including a motivation for this type of work). You then describe in detail the main design decisions of your render function, together with the results on the test sentences that you used (in Appendix). Report coverage and accuracy (think about how to measure this). In the discussion, you can explain problems that you encountered during grammar development and include an explanation for a certain behaviour of the grammar. You end your paper with a clear and brief conclusion, possibly hinting at further explorations of this task.

Deadline

11 June 2018, 4pm (= one week before the exam)

Submit your paper and the source code (in one file) through PointCarré (or by e-mail) *before the deadline*. You can use the forum for questions. Remember that extensions can only be granted if asked for enough in advance (at least one week!) and when you have a serious reason to miss the original deadline.

References

Babel2 Technical Documentation. https://fcg-net.org/tech-doc/