## Suggestions for the next stages of development of NLG system for generating headlines from creating briefs

Bogdan Babych b.babych@leeds.ac.uk

11 April 2019

The first idea for the NLG system has been to extract keywords from translation briefs, map these keywords to a part-of-speech template found in a corpus. Output sentences are produced and ranked from collocations of the keyword for each of the positions in the template, and further – from collocations of several top-ranked collocates for all positions in the template, with collocation scores being added for collocates generated for each position. This procedure is described in the document on Github: “Generating sentences from Part-of-Speech patterns using collocation field” https://github.com/bogdanbabych/colloc4nlg/blob/master/documentation/collocation-algorithm-v02.docx

Results for this approach have been generated on http://corpus.leeds.ac.uk/corpuslabs/lab201810cnlg/testKWsV3/

The proposed approach does not work as required. In my view there are two main problems:

1. The output is often not fluent, especially for longer templates. This may be due to the use of unigrams (0th order N-grams) as collocates, where the mechanism of cross-checking collocates across all positions in the template does not allow to ensure that a given unigram still fits with the context of the generated phrase or sentence.

This problem may be addressed in these ways:

1.1. Introducing a *language model* using higher order N-grams (e.g., up to 5-grams) into the generation process. The language model would inject correct word into the context even if collocation generation engine does not generate or rank the fitting item sufficiently high to show at the top of the list. The language model would compete with the generation model, and on balance the system should provide the most fluent, but still most relevant output given a keyword.

1.2. Introducing multiword collocations (as discussed during our project calls earlier). This approach would require

(a) generating a database of N-grams, e.g., up to 5-grams, or more generally – skip-grams, and calculating their frequencies in the advertising corpus;

(b) in run-time, for a given keyword in a concordance window of the length of a given part-of-speech template – calculating frequencies of all N- or skip-grams.

(c) for each N-/skip-gram in the concordance calculating collocation scores (log-likelihood, MI score, etc., given the N-/skip-gram’s overall frequency in corpus vs. its frequency in the concordance of the given keyword (these collocation scores measure how much the later frequency differs from the one expected by chance).

Multiword collocations would inject into PoS templates more fluent, naturally occurring chunks of text, but would still require a language model to smooth any possible “boundary friction” effects.

(The language model + multiword collocations have parallels in phrase-based statistical MT, which uses a language model and phrase-tables of N-grams).

2. The problem of generating expected, “boring” phrases, i.e., the ones which may have been seen by customers previous occasions and do not contain any unexpected, surprising or impressive words or phrases that would work for attracting attention to the advertised product.

E.g.: “*get an extra 10% off sale item*”

– <http://corpus.leeds.ac.uk/corpuslabs/lab201810cnlg/testKWsV3/cqp4nlg2cl_out4-additional.txt>

This is a more difficult problem to address, since collocations by their nature capture repetitive, standard language and do not provide a good model for creativity in language.

Some suggestions how this problem may be addressed:

2.1. Generating word embeddings (synonyms) of collocations for different positions in the template, which lead to a more creative language, e.g., see our papers:

<http://www.mt-archive.info/ACL-2007-Babych.pdf>

<http://www.mt-archive.info/Aslib-2007-Babych.pdf>

– if combined with a language model word embeddings could inject certain key words that on the one hand would be generally consistent with the context of a keyword, but on the other hand may come from different contexts and would be less expected within the generated sentence.

2.2. Finding better measures for capturing “innovative”, “affective”, “impressive”, “unexpected” yet linguistically coherent contexts, as an alternative to standard collocation scores. This can take a longer time and may require studies user-based studies, or possibly analysis of poetic language (e.g., finding measures which maximise certain scores in a corpus of poetry vs. a corpus of less innovative usage of language).

2.3. Introducing an ontology model to capture the structure of the subject domain, in addition to usage-based approach to NLG (generation from templates + collocations). There are standard approaches to NLG when sentences are produced from an underlying database (relational or RDF). In our case, an RDF database could be automatically populated or updated using a creative brief, but would provide additional information of what is standard, expected in the subject domain. In this way SPARQL queries to the database may reliably identify unique selling points or distinctive features of the advertised products, which should take priority in the process of the sentence generation, focussing the attention on the information that may be most relevant for customers.

The results show that the project is more complex than we initially expected, that simple approaches are not sufficient and may require a more systematic full-scale NLG system development, involving the language modelling and the ontology, but also (given the needs of the advertising industry) – good models for appellative, affective, unexpected usage of language that may inspire customers or increase productivity of creative teams. The project may require more resources and may involve risks for performance of the tool, since the “creative” language usage is not well understood and it is unknown how long it might take to produce a good model for it.

Also, from the coding perspective, it may be useful to move to a more productive development framework for the system, since Perl code has reached its limitations, and can no longer be effectively maintained and developed. It may be necessary to rewrite the code in Python in a more modular and observable way, which again would take time.