Realizing the Costs: Template-Based Surface Realisation in the GRAPH Approach to Referring Expression Generation

Ivo Brugman
University of Twente
The Netherlands
i.h.g.brugman@student.utwente.nl

Mariët Theune
University of Twente
The Netherlands
m.theune@utwente.nl

Emiel Krahmer
Tilburg University
The Netherlands
e.j.krahmer@uvt.nl

Jette Viethen
Macquarie University
Australia
jviethen@ics.mq.edu.au

Abstract

We describe a new realiser developed for the TUNA 2009 Challenge, and present its evaluation scores on the development set, showing a clear increase in performance compared to last year's simple realiser.

1 Introduction

The TUNA Challenge 2009 is the last in a series of challenges using the TUNA corpus of referring expressions (Gatt et al. 2007) for comparative evaluation of referring expression generation. The 2009 Challenge is aimed at *end-to-end referring expression generation*, which encompasses two subtasks: (1) *attribute selection*, choosing a number of attributes that uniquely characterize a target object, distinguishing it from other objects in a visual scene, and (2) *realisation*, converting the selected set of attributes into a word string. Our contributions to the previous Challenges focused on subtask (1), but this year we focus on subtask (2). Below, we briefly sketch how attribute selection is performed in our system, describe our newly developed realiser, and present our evaluation results on the TUNA 2009 development set.

2 Attribute selection

We use the Graph-based algorithm of Krahmer et al. (2003) for attribute selection. In this approach, objects and their attributes are represented in a graph as nodes and edges respectively, and attribute selection is seen as a graph search problem that outputs the cheapest distinguishing graph, given a particular *cost function* that assigns costs to attributes. By assigning zero costs to some attributes, e.g., the type of an object, the human tendency to mention redundant properties can be mimicked. For the TUNA Challenge 2009 we use the same settings as last year (Krahmer et al. 2008). The used cost function assigns a zero cost to attributes that are highly frequent in the TUNA corpus, while the other attributes have a cost of either 1 (somewhat infrequent) or 2 (very infrequent). The *order* in which attributes are added is also controlled: to ensure that the cheapest attributes are added first, they are tried in the order of their frequency in the TUNA (2008) training corpus. Using these settings, last year the GRAPH attribute selection algorithm made the top 3 on all evaluation measures (Gatt et al. 2008, Table 11).

3 Realisation

The main resource for realisation is a set of templates, derived from the human-produced object descriptions in the TUNA 2009 training data. To construct the templates, we first grouped the descriptions by the combination of attributes they expressed. For instance, in the domain of furniture references, all descriptions expressing the attributes colour, type and orientation were grouped together. This was done for all combinations of attributes. Next, for each description, parts of the word string were related to the attributes in the set. For instance, for the string "red couch facing left", we linked "red" to colour, "couch" to type, and "facing left" to orientation.[1] This provided us with information on how the attributes were expressed (e.g., by adjectives or prepositional phrases) and in which order they appeared in the word string. For each combination of attributes, the surface order that occurred most frequently was selected as the basis for a template. If multiple orderings were equally frequent, we chose the most natural-seeming one. This resulted in templates such as "the [colour] [type] facing [orientation]" for the attribute set {type, colour, orientation}.

During realisation, the templates are used as fol-

[1] This corresponds to the `ANNOTATED-WORD-STRING` nodes already present in the TUNA corpus. Unfortunately, various problems prevented us from automatically deriving our templates from those existing annotations.

lows. When a set of attributes is input to the realiser, it checks if there is a template matching this particular attribute combination. If so, the template is selected, and the gaps in the template are filled with lexical expressions for the attribute values. The words used to express the values are those that occurred most frequently in the training data for this particular template. If no matching template is found, a description is generated in a simple rule-based fashion, based on the realiser we used last year, but with improved lexical choices. For example, the old realiser always used the word "person" to express the type attribute in descriptions of people, whereas in the TUNA corpus "man" is used most frequently. We changed the realiser to reflect such human preferences.

Template construction for the furniture domain was fairly straightforward, resulting in 25 templates. In practice, only 13 of these are used. Since the GRAPH attribute selection algorithm adds the type and colour attributes to a description for free, these attributes are always selected, making any templates lacking them irrelevant given the current settings of the algorithm.

For the more realistic people domain, template construction was more complicated. For example, when the hairColour attribute is mentioned in human descriptions it can refer either to the hair on a person's head ("white-haired") or his beard ("with a white beard"). The attribute selection algorithm does not make this distinction, leaving it unclear which of the two realisations should be used when hairColour and hasBeard attributes are both to be included in a description. We solved this by simply using the expression that occurred most frequently in the training data for each attribute combination, even allowing hairColour to be mentioned twice if this happened in most human descriptions. Another problem is that many attribute combinations occurred only once in the training data, leading to a very large number (50+) of potential templates. We reduced this number in an ad hoc manner, by ignoring combinations involving attributes (such as hasHair) that are very unlikely to be selected given the current settings of the attribute selection algorithm. This approach left us with 40 templates in the people domain.

4 Evaluation

System performance is measured by comparing the generated word strings to the human descriptions in the TUNA development set, comprising 80 furniture and 68 people descriptions. The evaluation measures reported here are *mean edit distance* (MED), the mean of the token-based Levenshtein edit distance between the reference word strings and the system word strings, *mean normalised edit distance* (MNED), where the edit distance is normalised by the number of tokens, and cumulative BLEU 3 score. Table 1 summarizes our evaluation results. For comparison, we also provide the results obtained when using last year's simple realiser, which we reimplemented in Java.

	MED	**MNED**	**BLEU 3**
Furniture	4.94 (5.48)	0.48 (0.50)	0.27 (0.22)
People	5.15 (7.53)	0.46 (0.67)	0.33 (0.07)
Overall	5.03 (6.42)	0.47 (0.58)	0.30 (0.15)

Table 1: Results on the 2009 development set (between brackets are those using last year's realiser).

We see a clear improvement when we compare the performance of the new and the old realiser, in particular in the people domain. However, further evaluation experiments are required to determine whether the improvements are mostly due to our use of templates derived from human descriptions, or to the simple improvements in lexical choice incorporated in the rules used as fall-back in case no matching templates are found.

To further improve the realiser, we need to add templates for all remaining attribute combinations found in the corpus. This should not be difficult, as the set-up of the realiser allows easy creation of templates. It should also be easily portable to other languages; in fact we intend to explore its use for the realisation of referring expressions in Dutch.

References

Gatt, A., I. van der Sluis and K. van Deemter 2007. Evaluating algorithms for the generation of referring expressions using a balanced corpus. *Proceedings of ENLG 2007* 49-56.

Gatt, A., A. Belz and E. Kow 2008. The TUNA challenge 2008: Overview and evaluation results *Proceedings of INLG 2008* 198-206.

Krahmer, E., S. van Erk and A. Verleg 2003. Graph-based generation of referring expressions. *Computational Linguistics, 29(1)*, 53-72.

Krahmer, E., M. Theune, J. Viethen, and I. Hendrickx 2008. GRAPH: The costs of redundancy in referring expressions. *Proceedings of INLG 2008* 227-229.