UCM Submission to the Surface Realization Challenge

Pablo Gervás
Universidad Complutense de Madrid / Ciudad Universitaria, 28040 Madrid, Spain
pgervas@sip.ucm.es

1 Introduction

This document describes the surface realization solution submitted by UCM to the Surface Realization Challenge. The UCM submission operates over the shallow representation of the challenge input. This submission to the surface realization challenge relies on an old-fashioned surface realizer based on unification with a grammar. Because this surface realizer requires fully specified inputs, a complex conversion process is required from the challenge input to the data that needs to be provided to the realizer. Where the challenge input is underspecified, the conversion process must provide any information that is missing from the input.

2 The TAP SurReal Surface Realizer

The UCM submission to the surface realization challenge relies on the TAP framework previously used for the Referring Expression Generation Challenge 2008 (Gervás et al., 2008) and 2009 (Hervás and Gervás, 2009). TAP (Text Arranging Pipeline) is a Java API for generating simple fluent text from a Java application. TAP is not itself a surface realizer. Instead it relies on existing surface realizers to carry out its task. The current TAP implementation is configured to rely on the SurReal surface realizer. The SurReal (SURface REALizer) implementation provides a lightweight partial Java implementation of the surface realization mechanisms of FUF described in Elhadad (Elhadad, 1993). SurReal relies on a grammar which is unified with the input. This grammar follows the conventions of the FUF grammar in Elhadad (Elhadad and Robin, 1996), but it is currently a much more simplified version than the original in its scope.

The TAP SurReal combination employed here was developed to provide a light weight surface realizer for Spanish, with particular features intended to facilitate the generation of literary texts (such as explicit control of construct placement within the sentence). For the submission described here an initial sketchy grammar for English has been expanded as required to match the demands presented by the challenge input. In spite of the effort invested, coverage may still be significantly improved.

3 Converting the Challenge Input

The challenge input data for the shallow representation consists of unordered syntactic dependency trees. Each word and punctuation marker from the original sentence is represented as a node.

An initial stage of preprocessing is applied to eliminate nodes that are not useful to the surface realizer. These include most of the punctuation signs (colons and interrogation and exclamation marks are retained as they may provide relevant information). The additional marks for indicating which nodes fall inside quotations or brackets are also eliminated, as no method has been found to make use of the information they provide (in the limited amount of time alloted to ponder this issue).

Proper nouns that include nodes of the form `NAME_`$\star$ (with the $\star$ a number indicating their relative ordering) are collapsed into single `NNP` node with a string for the full name (in the order indicated).

The differences in nature between these depen-

dency trees and the input accepted by the surface realizer implies that the set of children nodes of any given node in a tree needs to be grouped into subsets that correspond to different subconstituents. Some of the information implicit in the surface form needs to be made explicit (such as agreement values for pronouns, or tense for clauses). Once this implicit information is explicit, the corresponding surface forms can be eliminated. These process is carried out by a set of hand-crafted tree rewriting rules. Rules rewrite, trim or relocate subtrees matching a given pattern, while respecting the rest of the tree (to ensure that a single abstract rule can cover a set of common cases in spite of ancillary local differences).

Due to the complexity of the task, at the time of writing only a limited set of such rules has been developed. Although an effort has been made to address the most generic constructions first, these rules fall short of covering the complete set of linguistic constructions available in the development data. The rules also fail to cover the full set of constructions that the realizer is capable of producing.

Although the shallow representation does have information on tense for specific nodes corresponding to verbs, the tense for each clause needs to be abstracted from the combination of tenses and the relative position of the various verb forms that make up the full verb phrase involved.

The explicit representation of pronouns in the shallow representation needs to be converted into the set of features that characterise them (person, number, gender).

Once the input trees have been rewritten to slimmer versions, a separate module converts them into suitable input for the realizer, using the TAP API.

Where the conversion process has failed to produce from the input successful data for the realizer, strings of the form "XXX*" has been introduced as place holders. Where the process resulted in no string, a place holder is required by the automation script, so the word "no" has been used.

4 Results over Development Data

The results obtained for the development data are reported on Table 1. These results are copied directly from the output of a version of the first automated script provided, adapted to run on a Windows machine.

BLEU	0.23791
BLEU (complete)	0.23791
Avg. BLEU	0.26100
Avg. BLEU (complete)	0.26100
NIST	2.59462
NIST (complete)	2.59462
Avg. NIST	4.48897
Avg. NIST (complete)	4.49331
METEOR	0.23061
Avg. METEOR	0.23061

Table 1: Single best results over development data

5 Discussion

The expected text provided with the development data is only used to provide feedback during the manual process of constructing the rewritting rules. This is a disadvantage with respect to alternative solutions capable of learning from the combination of input and expected text result.

The reported results constitute a measure of the coverage achieved by the input conversion process more than a measure of the capabilities of the realizer employed.

The TAP-SURREAL realizer provides rich features for controlling relative position of element within the sentence, however, as no information on relative position of elements in a sentence (other than for proper noun constructions) was available in the initial input data, the issue of relative position within the sentence has not been considered. The supplementary data with information on word position became available at too late a stage to be considered.

Acknowledgments

The research reported in this paper was research is partially supported by the Ministerio de Educación y Ciencia (TIN2006-14433-C02-01, TIN2009-14659-C03-01). Many thanks to Dominic Espinosa for his help in getting the automated evaluation scripts to run on a Windows machine.

References

M Elhadad and J Robin. 1996. An overview of SURGE: a reusable comprehensive syntactic realization component. Technical Report 96-03, Department of Computer Science, Ben Gurion University.

M Elhadad. 1993. FUF: The universal unifier. user manual, version 5.2. Technical Report CUCS-038-91, Columbia University.

P. Gervás, R. Hervás, and C. León. 2008. NIL-UCM: Most-Frequent-Value-First Attribute Selection and Best-Scoring-Choice Realization. In *Referring Expression Generation Challenge 2008, Proc. of the 5th International Natural Language Generation Conference (INLG'08)*.

Raquel Hervás and Pablo Gervás. 2009. Evolutionary and case-based approaches to REG: NIL-UCM-EvoTAP, NIL-UCM-ValuesCBR and NIL-UCM-EvoCBR. In *Proceedings of the 12th European Workshop on Natural Language Generation (ENLG 2009)*, pages 187–188, Athens, Greece, March. Association for Computational Linguistics.