Extracting Surface Realisation Templates from Corpora

Thiago D. Tadeu, Eder M. de Novais, Ivandré Paraboni
School of Arts, Sciences and Humanities - University of São Paulo (USP / EACH)
Av. Arlindo Bettio, 1000 São Paulo Brazil
E-mail: { thiagoo, eder.novais, ivandre} @usp.br

Abstract

In Natural Language Generation (NLG), template-based surface realisation is an effective solution to the problem of producing surface strings from a given semantic representation, but many applications may not be able to provide the input knowledge in the required level of detail, which in turn may limit the use of the available NLG resources. However, if we know in advance what the most likely output sentences are (e.g., because a corpus on the relevant application domain happens to be available), then *corpus knowledge* may be used to quickly deploy a surface realisation engine for small-scale applications, for which it may be sufficient to select a sentence (in natural language) that *resembles* the desired output, and then modify some or all of its constituents accordingly. In other words, the application may simply '*point to*' an existing sentence in the corpus and specify only the changes that need to take place to obtain the desired surface string. In this paper we describe one such approach to surface realisation, in which we extract syntactically-structured templates from a target corpus, and use these templates to produce existing and modified versions of the target sentences by a combination of canned text and basic dependency-tree operations.

1. Introduction

The present work has been developed in the context of a Q&A application under development, in which questions sent by students enrolled in an undergraduate project course will be answered semi-automatically by the system. In this application, answers are selected from a large database of standard replies (written by the professors in charge of the undergraduate project) to the most frequently asked questions made by the students and tailored to each particular context.

Our focus in this paper is the surface realisation subtask in Natural Language Generation (NLG) systems. Briefly, surface realisation concerns the production of surface strings from a given semantic representation, and it is considered one of the final stages in the language generation process in a standard NLG pipeline architecture (Reiter, 2007.)

One possible way of implementing a surface realisation engine is by making use of a grammar formalism, e.g., as in Bateman (1997). The use of a grammar guarantees robustness and wide coverage to surface realisation, but the level of detail required as an input may make a wide range of applications (namely, those that are not linguistically- motivated) difficult to adapt, which in turn limits the use of the NLG resources available.

The issue of input specification to the surface realisation task is a well-known research problem in the NLG field (e.g., Langkilde, 2000.) One possible way of simplifying the input requirements is by making use of surface realisation templates[1]. State-of-art template-based surface realisation systems such as YAG (McRoy et. al., 2003) rely on a relatively small number of template definitions and a powerful description language to provide fine-grained sentence specification. This enables more sophisticated NLG applications (i.e., those that are capable of providing detailed knowledge to fill in each template adequately) to take full advantage of template definitions and to have total control over the output text. For simpler applications, the challenge of input specification remains relatively unchanged.

There is however one particular case in which even knowledge-poor applications may benefit from NLG, namely, when we know in advance what the most likely output sentences are (e.g., because a corpus on the application domain happens to be available.) In these cases, we will argue that the existing knowledge can be used to quickly deploy a surface realisation component.

In what follows we describe one such approach to surface realisation, in which we extract syntactically-structured templates from a target corpus of standard answers to students' questions, and use these templates to produce existing and modified versions of the target sentences by a combination of canned text and basic dependency-tree operations. In doing so, we shall focus mainly on the general concept of surface realisation from corpus examples and the template hierarchy. The NLG approach proper will be described elsewhere.

The general principle that we adopt in this work is that for simpler applications it may be sufficient to select from corpora a sentence that *resembles* the desired output, and then modify some or all of its constituents accordingly. In other words, rather than specifying the sentence semantics in detail, the application may simply 'point to' an existing sentence in the corpus and specify only the changes that need to take place to obtain the desired surface string. This should arguably be much simpler (and of course much less flexible) than using a grammar-based surface realisation engine (e.g., Bateman, 1997) or even YAG-style templates (McRoy et. al., 1993.)

[1] For a comparison between template-based and other approaches to NLG, see for instance van Deemter et. al. (2005).

For instance, the application may pick from corpora a sentence such as “You may deliver your paper by Sunday” and specify that, leaving all other sentence constituents unchanged, the object to be realised in the sentence is “the results”. This will have the effect of producing (after certain agreement operations that need to take place in inflected languages such as Portuguese) the output “You may deliver the results by Sunday”. The overall effect is similar to what could be obtained by instantiating an appropriate template in systems such as YAG, but we believe that simpler NLG applications may benefit from a minimal input specification based on natural language (i.e., combining canned text and template values.)

The following Fig.1 illustrates the interaction between input semantic values (provided by the application), examples of sentence structures taken from corpora, and the output text produced by the surface realisation engine.

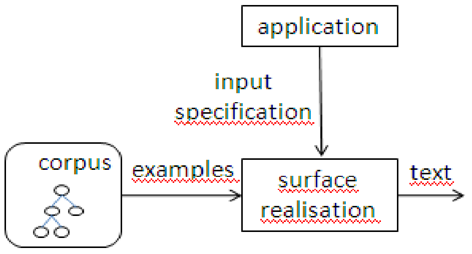

Figure 1: Surface realisation of an input semantic specification making use of corpus examples.

The reminder of this paper is structured as follows. Section 2 describes the kinds of template that we use in our system and the knowledge acquisition task. Section 3 discusses how the template hierarchy is used to generate text, and Section 4 presents our preliminary conclusions. Finally, Section 5 describes our ongoing and future work.

2. Current Work

The starting point of our work was the development of a database of surface realisation templates. Using a collection of emails sent to students in reply to their questions regarding an undergraduate project, we collected 597 sentences in Brazilian Portuguese and tagged/parsed using PALAVRAS (Bick, 2000). This collection was subsequently recast in XML format and constitutes our target corpus.

Each sentence in the target corpus is a template whose slots (discussed later) can be partially or totally filled-in to produce variations of the original sentence. However, due to a significant number of parsing and tagging errors, we notice that not all templates are actually functional for the purpose of sentence generation.

In our current work all template definitions were left as they are, that is, conveying the information provided by the tagging and parsing tools and including their occasional errors. We are however aware that since our approach relies heavily on the quality of the available templates, a real-world application will require these errors to be corrected.

Sentence templates may have up to three kinds of variable field: agent, patient and action. These constituents may be modified or replaced by the application by combining lower-order templates (e.g., for NPs and VPs) and additional canned text. Although clearly insufficient for wide-coverage, unrestricted text generation, in our application this level of variation represents a fine balance between ease of specification and flexibility of output expression, a point that we shall return to later.

As in other works in the field (e.g., Gatt & Reiter, 2009), we presently assume that the mappings from semantics to surface strings are to be provided by the underlying application. For testing purposes, however, we have extracted 1,548 unique instances of concept-to-string mappings from the target corpus, being 1,298 mappings from agent/patient entities to descriptions, pronouns and proper names, and 250 mappings from actions to VPs, even though not all of them are directly relevant to our application.

The following Figure 2 illustrates a sentence template with its variable fields (agent, action and patient) and their constituents represented as dependency-trees.

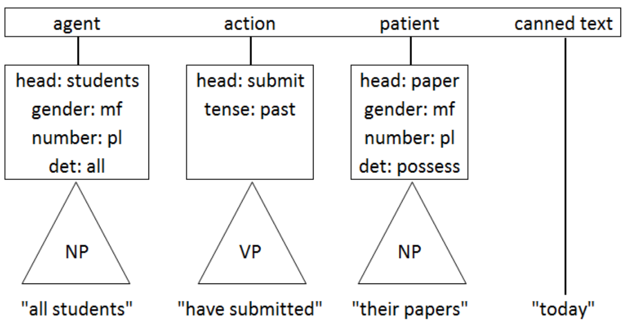

Figure 2: Sentence templates and variable fields.

The values provided by the application, if any, overwrite the default values of the template and, if necessary, basic agreement rules (e.g., between subject and object) are performed to ensure grammaticality. In our work this is accomplished with the aid of a database of inflections (Muniz et. al., 2005) and a thesaurus for the Brazilian Portuguese language.

Although based on a small set of examples, the combination of sentence, NP and VP templates with the ability to change individual template values may allow the application to generate a range of sentences that is still much wider than the target corpus. While this is by no means comparable to the flexibility of large-scale NLG systems, we believe that this may be sufficient for simple text-generating applications, and that its limitations (namely, the need to generate a sentence that does not resemble any other in the corpus) may be overcome by simply typing it in the text using natural language (which is arguably simpler than providing detailed instructions on how to combine templates) or by specifying the new sentence from scratch (in which case more input knowledge will be required.)

3. Template-based Generation

In this section we will provide an overview of the text generation task. Details about the current state of the NLG system will be described elsewhere.

Using the template definitions described in the previous section, we designed a simple corpus-based surface realisation engine. Our surface realisation module can be used in two ways: first, in our proposed template-based approach based on corpus examples, the surface realisation engine takes as an input a template id (i.e., a sample structure with default values for the output sentence) and, optionally, additional parameters representing the alternative semantics of its agent, patient and action constituents, including gender, number, tense, mode, reference type (e.g., definite vs. indefinite etc.) etc.

Alternatively, our surface realisation engine also offers a number of pre-defined templates for common sentence structures (e.g., in the form NP VP NP.) These standard templates are not based on corpus examples, and they are implemented as a means to add flexibility to the system, and to make it more usable. Although in principle defeating the concept of generation from corpus examples, we notice that standard sentence structures are numerous in many applications, and using a standard template may in some cases be simpler than selecting and modifying a corpus sentence.

A complete example works as follows: in a full canned text approach, the underlying application may simply select the required template id to produce the desired output verbatim as in (a); with some additional knowledge available, the application may change some aspects of the output sentence (e.g., the agent and patient fields) as in (b); finally, with a nearly-full semantic specification as an input (e.g., including new values for the action field), the original structure may change even further as in (c), in which case only the non-terminal nodes of the dependency-tree were preserved. Thus, we have used the template specification in (a) simply as an example to produce an entirely different sentence as in (c).

(a) [You]$_{\text{agent}}$ [have not finished]$_{\text{action}}$ [your thesis]$_{\text{patient}}$

(b) [The students]$_{\text{agent}}$ [have not finished]$_{\text{action}}$ [their homework]$_{\text{patient}}$

(c) [The students]$_{\text{agent}}$ [will complete]$_{\text{action}}$ [their homework]$_{\text{patient}}$

Finally, we notice that not all dependency-tree replacements are possible in practice, which bears a number of consequences to the overall system behaviour. For example, our application often produces imperative statements that (in Portuguese) do not convey an explicit subject, as in “Please do your homework”. In these cases, it is not clear what it means for the NLG application to be requested to insert an agent constituent where there is none.

A similar situation may arise if an action that is incompatible with the current template definition is requested to replace the main verb in the structure. For example, it is unclear whether the system should allow the main verb in the template “You *are* very clever” to be replaced by, e.g., “buy”, in which case a ungrammatical sentences such as “#You *buy* very clever” would be produced.

In our current work, a request of the first kind (i.e., inclusion of an agent in a sentence without a subject) is simply ignored by the surface realisation module, and no change in the original structure takes place. In the second case (invalid verb replacement), however, the system does proceed with the replacement of the verb tree, the underlying assumption being that the application is responsible for what is provided as an input. In either case, however, remains the question of how far a select-and-modify approach can go without undesirable results. We presently assume that more research on this issue is still required.

4. Conclusion

In this paper we have proposed a simple approach to surface realisation based on the (re)use of syntactically-structured templates acquired from corpora. Although not nearly as flexible as a full NLG approach, our system may represent a straightforward solution to the problem of input specification, which in our case is simply based on natural language. Our corpus-based approach is able to generate single sentences from an input conveying various degrees of semantic knowledge, which may be suitable to a wide range of NLG applications that are able to provide more or less detailed linguistic knowledge as an input.

Although the present system is considerably less sophisticated than, e.g., a full template-based approach as YAG (McRoy et. al., 2003) or a grammar-based approach as KPML (Bateman, 1997), we notice that the level of knowledge representation used as an input seems well-balanced for our particular application. In other words, had the surface realisation engine required more fine-grained input, it would probably be difficult to adapt our application to it. On the other hand, had we defined an even simpler input specification, the surface realisation would amount to little more than canned text, which would not be sufficiently flexible for our present needs.

5. Future work

Our systems is currently functional at a prototype level only, and a first assessment of the present approach is underway. More specifically, we are building a reference set of manually-written sentences derived from the template collection, each of them conveying a number of pre-defined modifications. For example, we take a particular sentence template from the corpus, and replace its agent constituent manually using another (also pre-defined) content value. Once this task is finalised, the same set of sentences will be generated by the system, and both system and reference sets will be compared against each other using standard evaluation metrics such as edit-distance, and BLEU/NIST (Papineni et. al., 2002; NIST 2002.)

Besides the evaluation work, the following improvements are considered: first, we are currently expanding the possible lexical choices by making use of a thesaurus and a language model to select the most likely output as, e.g., in Langkilde (2000); second, the mappings from semantic-concepts to surface strings still need to be revised and adapted to our current domain (questions about students' undergraduate projects;) finally, we intend to automatically acquire new sentences from the target corpus without any off-line pre-processing (e.g., parsing and template extraction.)

6. Acknowledgements

The authors acknowledge support by FAPESP and CNPq.

7. References

Bateman, J.A. (1997) Enabling technology for multilingual natural language generation: the KPML development environment. Natural Language Engineering, 3(1):15–55.

Bick, E. (2000) The parsing system PALAVRAS: automatic grammatical analysis of Portuguese in a constraint grammar framework. *PhD Thesis*, Aarhus University.

Gatt, A. and E. Reiter (2009) SimpleNLG: A realisation engine for practical applications. *Proceedings of ENLG-2009*, Athens, Greece.

Langkilde, Irene (2000) Forest-based statistical sentence generation. 6th Applied Natural Language Processing Conference and 1st Meeting of the North American Chapter of the Association of Computational Linguistics (ANLP-NAACL'00), pp. 170–177.

McRoy, S., S. Channarukul and S. S. Ali (2003) An augmented template-based approach to text realization. *Natural Language Engineering* 9 (4) pp. 381–420. Cambridge University Press.

Muniz, M. C., Laporte, E., Nunes, M.G.V (2005) UNITEX-PB, a set of flexible language resources for Brazilian Portuguese. III Information and Language Technology Workshop (TIL-2005).

NIST (2002) Automatic Evaluation of Machine Translation Quality using n-gram Co-occurrence Statistics".*www.nist.gov/speech/tests/mt/doc/ngram-study.pdf*

Papineni, S., T. Roukos, W. Ward, and W. Zhu (2002) Bleu: a method for automatic evaluation of machine translation. *Proceedings of ACL-02*, pp. 311–318.

Reiter, E. (2007) An Architecture for Data-to-Text Systems. *Proceedings of ENLG-2007*, pp. 97-104.

van Deemter, K., Emiel Krahmer and Mariët Theune (2005) Real versus template-based NLG: a false op-position? Computational Linguistics 31(1).