Text Generation for Brazilian Portuguese: the Surface Realization Task

Eder Miranda de Novais **Thiago Dias Tadeu** **Ivandré Paraboni**
University of São Paulo - School of Arts, Sciences and Humanities (USP-EACH)
Av. Arlindo Bettio, 1000, Ermelino Matarazzo
São Paulo, Brazil - 03828-000
eder.novais@usp.br tdtadeu@gmail.com ivandre@usp.br

Abstract

Despite the growing interest in NLP focused on the Brazilian Portuguese language in recent years, its obvious counterpart – Natural Language Generation (NLG) – remains in that case a little-explored research field. In this paper we describe preliminary results of a first project of this kind, addressing the issue of surface realization for Brazilian Portuguese. Our approach, which may be particularly suitable to simpler NLG applications in which a domain corpus of the most likely output sentences happens to be available, is in principle adaptable to many closely-related languages, and paves the way to further NLG research focused on Romance languages in general.

1 Introduction

Data-to-Text Natural Language Generation (NLG) systems produce text or speech from a given non-linguistic input. Systems of this kind usually follow a pipelined architecture (Reiter, 2007) comprising data interpretation, document planning, sentence planning and surface realization tasks. In this work we discuss the latter, that is, the task of producing surface word strings from a non-linguistic input specification.

Existing approaches to surface realization may vary greatly in their input requirements and, consequently, in the level of control over the output text. On the one hand, more sophisticated, grammar-based surface realization systems such as KPML (Bateman, 1997) allow maximum flexibility and productive coverage. These advantages, however, are only useful if the underlying application is capable of providing a detailed semantic specification as an input to the surface realization module in the first place.

As an alternative to surface realization grammars, NLG systems may also rely on *template-based* surface realization, that is, the use of predefined structures with a number of variable fields (or slots) to be filled in with values provided by the application. For a comparison between templates and other approaches to NLG, see for instance van Deemter et. al. (2005).

Adapting an existing application to a template-based realization system is usually much simpler than in a grammar-based approach. Yet, in order to take full advantage of template definitions and to obtain a degree of control over the output text that is comparable to what a grammar-based system would allow, it is still necessary to master the use of templates and their rules to fill in each slot adequately.

The problem of input specification to surface realization has been discussed at length in the literature in the field - see for example Langkilde (2000) – and we of course do not dispute that more sophisticated NLG systems will require a detailed input specification. However, given that the available semantics may not be provide in this level of detail, in this paper we discuss an alternative that

Proceedings of the NAACL HLT 2010 Young Investigators Workshop on Computational Approaches to Languages of the Americas, pages 125–131, Los Angeles, California, June 2010.

may be suitable to simpler applications, namely, those cases in which it is known in advance what the most likely output sentence structures are, for example, because a corpus on that particular domain happens to be available. In these cases, we will argue that it may be possible to take advantage of the available knowledge to quickly deploy a surface realization component based on existing corpora.

The underlying assumption in our work is that there are simpler NLG applications for which it may be sufficient to select a sentence that *resembles* the desired output, and then modify some or all of its constituents as needed to achieve the desired output. For instance, an application that is not linguistically-oriented may produce its output results as natural language text by selecting a standard imperative sentence as in "Please reply to this message" and, leaving all other sentence constituents unchanged, specify that the action to be realized in the output is "delete", and that its patient object is "file". This will have the effect of producing the output sentence "Please delete this file".

In this introductory work we intend to outline our ongoing efforts to develop one such approach to surface realization for the Brazilian Portuguese language. In doing so, we shall focus on the general principles that guide our research, leaving much of the theoretical details to be discussed elsewhere. The present work has been developed within the context of a query-and-answer application under investigation, in which questions sent by undergraduate students enrolled in a particular course will be matched to existing entries in a large database of standard replies written by the professors in charge to the most frequently asked questions made by the students, and tailored to each particular context accordingly. Details of this particular application will not be dealt with in this paper either.

The reminder of this paper is structured as follows. Section 2 briefly discusses related work on surface realization; Section 3 provides an overview of our system's architecture; Sections 4 describes the extraction of syntactically-structured templates from a target corpus and Section 5 presents the current features of our template-based surface realization engine. Finally, Section 6 draws preliminary conclusions and describes ongoing work, and Section 7 hints at possible collaboration with the wider NLP research community in Latin America and elsewhere.

2 Related work

Mapping an application semantics to surface strings usually involves the use of surface realization grammars or similar resources, which can be either built manually (e.g., Bateman, 1997) or acquired automatically from a corpus (Ratnaparkhi, 2000; Zhong & Stent, 2005; DeVault et. al., 2008).

The surface realization task proper can be divided into two relatively independent procedures: a domain-dependant mapping from the application semantics onto linguistic structures (including, e.g., lexical choice), and a language-oriented task of linearization. As pointed out in Gatt & Reiter (2009), most of the existing systems tend to perform both tasks, but in some cases they focus on the latter, assuming that all lexical choices and other domain-dependent decisions have already been made. This is the case for example of SimpleNLG (Gatt & Reiter, 2009), a surface realization engine implemented as a Java library for sentence linearization.

Central to the development and use of a surface realization system is the kind of input specification that will be expected from the application. In order to take full advantage of grammar-based surface realization, it is usually necessary to provided detailed linguistic knowledge as an input. This is the case, for example, of a number of corpus-based approaches to grammar acquisition, which may take logical forms as an input (e.g., Smets et. al., 2003; Zhong & Stent, 2005; Marciniak & Strube, 2005). The Amalgam system, for instance (Smets et. al., 2003), takes as an input a graph conveying fixed content words (lemmas) and detailed linguistic information such as verb tense and mode, gender, number and definiteness of all its constituents, and additional semantic features (e.g., 'human', 'animated' etc.)

Detailed input specification as required in grammar-based surface realization is however often unavailable from the semantics of the application. As an alternative, template-based surface realization makes use of predefined structures (e.g., syntactically-structured sentence templates) with slots to be filled in with values provided by the application. A prominent example of template-based surface realization system is YAG (McRoy

et. al., 2003), which may accept both feature structures and propositional semantics as an input. The following is an example of input feature structure in YAG, taken from McRoy et. al. (2003). In this example, the structure represents the fact that a discourse subject (George) performs an act (understand) on a particular object (a book), in which both subject and object happened to be realized as pronouns as "He understands it".

```
((template clause
 (process "understand")
 (agent   ((template noun-phrase)
           (np-type PROPER)
           (head "George")
             (gender MASCULINE)
             (pronominal YES)))
 (object ((template noun-phrase)
          (head "book")
          (pronominal YES))) )
```

Input feature structure in YAG.

The input requirements of a template-based surface realization system are obviously much simpler – and more likely to be available from the application – than a full set of linguistic instructions on how to generate the desired output. Still, in this work we would like to produce surface strings using even less knowledge, namely, by using sentence-level templates extracted from a domain corpus as a basis to generate original and modified versions of the corpus sentences.

We will refer to this as an example-based approach to surface realization[1], although this is not to be mistaken for example-based learning techniques to perform automatic grammar induction as in DeVault et. al., (2008), or other forms of grammar acquisition as in Zhong & Stent (2005). Our work is more related to Ratnaparkhi (2000) in the sense that we also use a large collection of generation templates for surface realization, but still distinct in that we intend to generate text from minimal input.

3 Project Overview

Template-based surface realization systems such as YAG (McRoy et. al., 2003) make use of a relatively small number of template definitions and some kind of descriptive language to provide fine-grained input sentence specification with flexibility and wide coverage. However, if a corpus on the application domain happens to be available, and assuming that the corpus sentences resemble those that we intend to generate, then it may be convenient (at least for applications that are not linguistically-motivated in the first place) to simply use the corpus sentences as examples, and allow an input specification that makes explicit only the changes that need to take place to convert the selected example into the desired output.

For example, in order to produce the sentence "He understands it" we may select an example such as "People will understand it" from the corpus, and then redefine its agent head type as a pronoun, and its action tense as present. The difference may not seem so dramatic if compared to, e.g., an input specification to YAG, but it will obviously grow as more complex sentence structures are considered.

If the selected example differs greatly from the target sentence, then a large number of modifications will have to take place, and in that case our example-based approach may not seem very useful. On the other hand, if the corpus is representative of the sentences that are likely to be generated, then little or no additional modifications will be required, in which case new sentences may be generated indeed from a minimally specified input. In either case, we notice that since the examples are represented directly in natural language in the corpus, new instances can be easily added to expand the system coverage.

In our present approach to the surface realization task, syntactically-structured templates are selected from a target corpus on the application domain and used as a basis to produce original and modified versions of the corpus sentences by a combination of canned text and basic dependency-tree operations. Each sentence in the target corpus makes a sentence template in which the agent, patient and action constituents may be modified or replaced by the application by combining lower-order templates (e.g., for NPs and VPs), and new sentences may be supported by adding the corresponding examples directly to the corpus.

Our current work can be divided into two main tasks: the extraction of syntactically-structured templates from corpora and the actual development of the surface realization engine. The following sections 4 and 5 discuss each of these tasks in turn.

[1] Perhaps 'select-and-modify' would be closer to our current purposes.

4 Template Extraction

Using a collection of emails sent to undergraduate students by their professors in reply to their most frequent questions regarding a particular project, we developed a database conveying 597 instances of surface realization templates for Brazilian Portuguese NLG as follows.

After sentence segmentation, the corpus was tagged and parsed using PALAVRAS (Bick, 2000). A number of critical parsing errors were removed, and thus we arrived at a set of 578 sentence-level templates represented in XML format.

In our example-based approach to surface realization we consider two kinds of structure: sentence and constituent templates. Sentence templates are high-level representations of the sample sentences taken from our target corpus, and they contain a number of variable fields (the constituents) to be filled in with application data (in most cases having an agent, action and patient fields.)

Everything else within the sentence is simply canned text as seen in the corpus, and cannot be modified by the application. In other words, if the application needs to generate a sentence that differs from the template in any constituent other than its NPs and VPs, it is necessary to define a new template by adding a new example to the corpus.

Sentence templates are highly redundant in the sense that many of them keep a similar syntactic structure in which only the surrounding text might change significantly. For example, many sentence templates in our domain represent a simple piece of advice in the form agent + action followed by some canned text, as in "You should enroll by Friday" and "All smokers are supposed to quit by the end of the month".

Although we could have defined a smaller (and more flexible) set of templates by generalizing over these structures, in practice this would increase the complexity of the required input (e.g., with the addition of a 'time' field to a common template to be shared by both examples above.) As mentioned in the previous section, we intend to keep input specification as simple as possible (i.e., in natural language format) by allowing the target sentences to be specified directly in the corpus.

The contents of the variable fields in a sentence template act as default values for the surface realization algorithm, and they may be changed individually (e.g., by setting a different tense or gender value for a particular field) or replaced by another constituent template entirely. We notice that default values are acquired automatically from corpora, i.e., they do not need to be hard-coded as in McRoy et. al., (2003).

Unlike sentence templates, constituent templates are not extracted from corpora. Instead, constituents are dependency-trees generated by a small set of grammar rules that covers the instances of VPs and NPs found in our corpus, including support to relative clauses and the most common forms of PP attachment. The choice for a grammar representation for the more fine-grained constituents was mainly motivated by the need to achieve wider coverage and to support linguistic variation beyond what the actual phrases found in the corpus would allow. In doing so we are able to fill in sentence templates with phrases of arbitrary complexity, as in the NP "You should enroll by the end of *the month in which you are expected to complete your current assignment*", and not simply using those NPs found in the target corpus.

The set of mappings from domain concepts to their dependency-trees (i.e., constituent templates) makes a dictionary of realizations in the application domain. As in related work in the field (e.g., Gatt & Reiter, 2009), we presently assume that the actual mappings are to be provided by the application.

Concept-to-strings mappings are usually handcrafted, but may also be acquired automatically from corpora, as in Bangalore & Rambow (2000). For testing purposes, we have extracted 1,548 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 many of them will not be of practical use from the point of view of our intended application.

5 Surface Realization

Using the template definitions from the previous section, we designed a simple corpus-based surface realization component for our ongoing project.

Our surface realization module is currently able to accept as an input a template id (to be taken as a sample structure with inherited default values for the output sentence) and, optionally, parameters representing the alternative semantics of its agent,

patient and action constituents. Alternatively, it is also possible to specify a sentence from scratch (that is, without using any existing template as a basis) in a standard NP VP NP format. The latter choice was added to the system as we noticed that simpler sentence structures may be specified more conveniently in this way, as opposed to looking up an example in the corpus. In our project, this is the case of short reply sentences as in "Yes, of course", "Thank you" and others, in which there is hardly any point in selecting a template from the corpus and then commanding the required changes.

The underlying application selects a target template and provides a set of values to fill in the template variable fields. These input values overwrite the default values provided by the template (that is, those values that were inherited from the corpus data) and adjusted by basic agreement rules to reestablish grammaticality if necessary, as we will discuss later.

The currently supported variable fields for NPs are determiner type, gender, number, person, determiner lemma, pre and post modifiers, the NP head, an attached pp-list and relative clause (which may recursively convey NPs within themselves.) As for VPs, the variable fields are VP type (finite vs. infinite etc.), person, mode, verb type, verb tense and adverbial modifiers. Verbal gender and number are not specified directly but simply inherited from the subject's own data to avoid a possibly conflicting input specification.

The most obvious limitation to this kind of approach is the case in which there is a need to generate a sentence that does not resemble any example in the corpus at all. Yet again, we notice that this difficulty may be overcome by simply adding a natural language example directly to the corpus, a method that is arguably simpler than providing detailed instructions on how to select and combine template structures in a traditional template-based approach, and even simpler than providing a full sentence specification in grammar-based surface realization.

The following is a complete example of how the example-based approach is expected to work. In its simplest form, the application may select the required template to produce the desired output verbatim as in (a); with some extra knowledge available, the application may also change some of the values of the variable template fields as in (b); finally, with even more complete linguistic knowledge available, the original structure may be changed even further as in (c), in which case only the original sentence structure remained (besides the canned text component "on Friday").

	Input	**Expected output**
(a)	template #17	[You]$_{agent}$ [should deliver]$_{action}$ [your results]$_{patient}$ on Friday.
(b)	template #17, patient=essay, action=not_complete	[You]$_{agent}$ [did not complete]$_{action}$ [your essay]$_{patient}$ on Friday.
(c)	template #17, agent=teacher, determiner=possess, action=give, tense=future, patient=talk, determiner=indefinite	[Our teacher]$_{agent}$ [will give]$_{action}$ [a talk]$_{patient}$ on Friday.

Table 1. Examples of (semantic) input and expected (surface text) output.

Depending on the changes in the constituent values requested by the application, a number of agreement rules may be invoked to re-establish fluency and grammaticality. In our work this is aided by a Brazilian Portuguese lexicon presented in Muniz et. al. (2005) and a thesaurus. For example, if a sentence template as (d) below is selected, and then the value of the agent head field is changed to represent a singular concept as in (e), agreement rules are required to modify the verb number as in (f).

(d) [All students]$_{agent}$ [have submitted]$_{action}$ [their papers]$_{patient}$

(e) [Your teacher] $_{agent}$ [#have submitted]$_{action}$ [their papers]$_{patient}$

(f) [Your teacher] $_{agent}$ [has submitted]$_{action}$ [their papers]$_{patient}$

Table 2. An original example (d) reused with a new agent head value (e) and agreement (f).

More complex or fine-grained dependencies (e.g., the anaphoric reference 'their' in Table 2

above) are not currently implemented. One possible approach to this is a standard generate-and-select approach to NLG as in Langkilde (2000), Oh & Rudnicky (2000) and others. More specifically, we may over-generate all possible realization alternatives and then use a statistical language model to select the most likely output. In our work we intend to apply a similar approach also to handle the lexical choice task, i.e., by selecting the most likely wording for each concept based on a language model.

6 Discussion

In this paper we have described a simple approach to surface realization based on the reuse 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 support only less detailed input specification.

Much of the present work is however to be regarded as tentative. One major issue that is yet to be discussed is how far we can go with an example-based approach to surface realization without compromising the quality of the output text. For instance, it is not clear what it means for the NLG system if the application selects a sentence template that (in Portuguese) does not have a subject field (e.g., "Please send it now") and then attempts to specify a subject. A similar conflict arises, for example, if the application specifies an action that is semantically incompatible with the selected template, in which case the output sentence could become ungrammatical. In both cases, we believe that more research is still needed.

Being currently functional at a prototype level only, our system is undergoing a number of improvements. First, we are expanding the possible lexical choices by making use of a thesaurus, and then we intend to use a language model to handle synonymy.

Second, the mappings from semantic concepts to surface strings still need to be revised and adapted to the domain (questions and answers about students' undergraduate projects) in order to deploy a fully functional application.

Finally, template selection needs improvement to allow for a truly minimal input specification in an application-friendly fashion.

With these tasks accomplished, we will be able to attach a surface realization component to our ongoing Q&A project and generate context-sensitive replies to students' most frequent questions.

7 Final Remarks

In the context of the NAACL-HLT Young Investigators Workshop on Computational Approaches to Languages of the Americas, there are a number of ways in which our work could benefit from cooperation with researchers in Latin America, and also help the development of NLP research in these countries.

At the current stage, our work still relies heavily on a Portuguese parsed corpus and grammar, which may be seen being of limited interest outside the Brazilian NLP research community. However, given the close relation between Portuguese and other languages spoken in the region (e.g., Spanish and its variations), we believe that it would be a rewarding experience to adapt similar language resources (e.g., sentence templates, phrase grammars etc.) that have been developed elsewhere, and use these resources to deploy a multilingual NLG application to validate our current approach.

Beyond the usefulness to the research communities involved, we would expect that this kind of cooperation would be an effective means of sharing costs and spreading the interest in NLG research across the region, and a much-needed motivation for young researchers to join the field.

Acknowledgments

The authors acknowledge support by FAPESP and CNPq. We are also thankful to the anonymous reviewers of the original submission, and to the organizers of the NAACL-HLT Young Investigators Workshop on Computational Approaches to Languages of the Americas for the travel award given for this presentation.

References

Bangalore, S. and O. Rambow (2000) Corpus-based lexical choice in natural language generation. Proceedings of the 38th Meeting of the ACL, Hong Kong, pp. 464-471.

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.

DeVault, David, David Traum and Ron Arstein (2008) Practical Grammar-Based NLG from Examples. Proceedings of the 5th International Natural Language Generation Conference (INLG-2008) Columbus, USA.

Gatt, Albert and Ehud Reiter (2009) SimpleNLG: A realization engine for practical applications. Proceedings of the European Natural Language Generation workshop (ENLG-2009.)

Langkilde, Irene (2000) Forest-based statistical sentence generation. Proceedings of the 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.

Marciniak, T. and M. Strube (2005) Using an Annotated Corpus As a Knowledge Source For Language Generation. Proceedings of the Corpus Linguistics'05 Workshop Using Corpora for NLG (UNNLG-2005), pp. 19-24.

McRoy, Susan, Songsak Channarukul and Syed 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. Proceedings of the III Information and Language Technology Workshop (TIL-2005).

Oh, A. and A. Rudnicky (2000) Stochastic language generation for spoken dialogue systems. Proceedings of the ANLP-NAACL 2000 Workshop on Conversational Systems, pp. 27–32.

Ratnaparkhi, A. (2000) Trainable methods for surface natural language generation. Proceedings of ANLP-NAACL 2000, pp.194–201.

Reiter, E. (2007) An Architecture for Data-to-Text Systems. Proceedings of the European Natural Language Generation workshop (ENLG-2007), pp. 97-104.

Smets, M., M.Gamon, S.Corston-Oliver and E. Ringger (2003) French Amalgam: A machine-learned sentence realization system. Proceedings of the TALN-2003 Conference, Batz sur-Mer,

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

Zhong, Huayan and A. J. Stent (2005) Building Surface Realizers Automatically from Corpora. Proceedings of the Corpus Linguistics'05 Workshop Using Corpora for NLG, pp. 49-54.