Visual Development Process for Automatic Generation of Digital Games Narrative Content

Maria Fernanda Caropreso[1] **Diana Inkpen**[1] **Shahzad Khan**[2] **Fazel Keshtkar**[1]

[1]University of Ottawa
{caropres,diana}@site.uottawa.ca
akesh081@uottawa.ca

[2]DISTIL Interactive
s.khan2@distilinteractive.com

Abstract

Users of Natural Language Generation systems are required to have sophisticated linguistic and sometimes even programming knowledge, which has hindered the adoption of this technology by individuals outside the computational linguistics research community. We have designed and implemented a visual environment for creating and modifying NLG templates which requires no programming ability and minimum linguistic knowledge. It allows specifying templates with any number of variables and dependencies between them. Internally, it uses SimpleNLG to provide the linguistic background knowledge. We tested the performance of our system in the context of an interactive simulation game. We describe the templates used for testing and show examples of sentences that our system generates from these templates.

1 Introduction

Natural Language Generation (NLG) is the process of constructing outputs from non-linguistic inputs (Bateman, 2002) (Dalianis, 1996) (Reiter and Dale, 2000).

NLG systems are useful in systems in which verbal or textual interaction with the users is required, as for example Gaming, Robotics, and Automatic Help Desks. Using NLG systems instead of manually authored sentences would enable the software to adapt the expressed messages to the context of the conversation, and express past and future actions that may form this interaction.

However, the use of the available NLG systems is far from simple. The most complete systems often require extensive linguistic knowledge. Some systems also require programming knowledge. This knowledge cannot be assumed for the content and subject matter experts who are members of a development team. However, these individuals do need to interact with the NLG system in order to make use of the message generation capability to support their product development efforts. It is then necessary to provide them with an environment that will allow them to have access in a simpler way to the features they need of a specific NLG system.

There are two widely adopted approaches to NLG, the 'deep-linguistic' and the 'template-based' (van Deemter et al., 2005). The deep-linguistic approach attempts to build the sentences up from a wholly logical representation. The template-based NLG systems provide scaffolding in the form of templates that contain a predefined structure and perhaps some of the final text.

SimpleNLG is an NLG system that allows the user to specify a sentence by giving its content words and its grammatical roles (such as subject or verb). SimpleNLG also permits the user to specify several features for the main verb, such as: tense (present, past or future); whether or not it is subjective, progressive, passive or perfect; whether or not it is in interrogative form; whether or not it is negated; and which, if any, modal to use (i.e. could, must).While some of these features affect only the verb, others affect the structure of the whole sentence, as for example when it has to be expressed in the passive voice.

SimpleNLG is implemented as a java library and it requires java programming knowledge to be used. Because of the programming nature of SimpleNLG, it allows the user to define flexible templates by using programming variables in the sentence specification. The variable parts of the templates could be filled with different values. When templates are defined using SimpleNLG they keep all the functionality of the NLG system (for example, being able to modify the verb features or the output format, and making use of the grammatical knowledge), while also allowing for the variable values to change.

We have designed an environment that provides simple access to the use of the SimpleNLG system in order to generate sentences with variable parts or templates. We developed this NLG Template Authoring Environment guided by the need of templates required for generating content for digital-based training games at DISTIL Interactive[1]. An early prototype of the tool, with a text-only interface, is presented in (Caropreso et al., 2009).

In training games the player is typically presented with challenging situations and is encouraged to practice different strategies at dealing with them, in a safe, virtual environment. Through tips and feedback, the player develops an understanding of the problem and what are the successful ways of confronting it (French et al., 1999).

In training games there is usually an explosion of possible scenarios and situations. The narrative should ideally reflect the past events and decisions taken. The considerable amount of textual information required in order to keep the feedback consistent with the updated narrative can be a burden on the game designers. It is then necessary to include templates that statically provide the basic information, combined with variable parts that adapt the narrative to the circumstances.

The goal of the NLG Template Authoring Environment was to provide the game content designers with an accessible tool they could use to create and manipulate the NLG templates, and thus generate sentences that would support the narrative progression of the game.

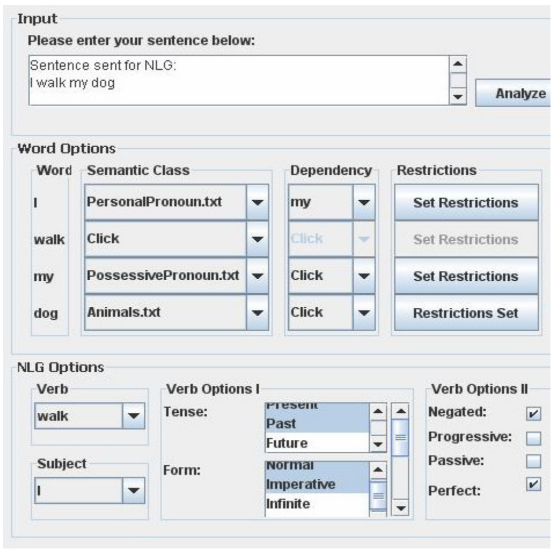
In the rest of this paper we describe our NLG Template Authoring Environment, its design, implementation and capabilities. We describe the templates that we used to test the system and we explain the user's knowledge required in order to create them. We finish the paper presenting our conclusions and future work.

2 Template Authoring Environment

The NLG Template Authoring Environment asks for a model sentence and allows the user to mark the sections that are variable. For each variable indicated, the user has to specify its type (i.e., personal pronoun, possessive pronoun, Employee_type) and which values of that type are allowed (i.e., all personal pronouns, or only "she" and "he"). Additionally, the user can also indicate dependencies between variable elements and information for the verb (i.e., tense, form, modals). The system then shows the user all the possible sentences that could be generated from the given template by calculating all the possible combinations of variable values that respect the specified dependencies and follow the verb selections. The user can then refine the template by changing the given example or the specified variables, dependencies or verb options, in order to adjust the generated sentences to the needs of the game.

The NLG Template Authoring Environment has been implemented in Java. The SimpleNLG library was used to automatically generate correct sentences and provide the user with the possibility of exploring different attributes to the verb. It has a user-friendly intuitive graphical interface, part of which is shown in Figure 1.

Figure 1: Graphical Interface



Input

Please enter your sentence below:

Sentence sent for NLG:
I walk my dog

Word Options

Word	Semantic Class	Dependency	Restrictions
I	PersonalPronoun.txt	my	Set Restrictions
walk	Click	Click	Set Restrictions
my	PossessivePronoun.txt	Click	Set Restrictions
dog	Animals.txt	Click	Restrictions Set

NLG Options

Verb: walk

Subject: I

Verb Options I

Tense: Present, Past, Future

Form: Normal, Imperative, Infinite

Verb Options II

Negated: ✓
Progressive:
Passive:
Perfect: ✓

After entering an example sentence and clicking on Analyze, the user indicates that a section is variable by giving a type or semantic class to the word in that section. The values of a semantic class are stored in a text file, which allows the user to create new semantic classes as needed. These files contain all the possible values and their respective syntactic information (person, number and gender) which will be used for agreement with the verb and for dependency between variables purposes. Restrictions to the values that a variable can take are also indicated

[1] http://www.distilinteractive.com/

through the graphical interface. Dependencies can be indicated only between already declared variables. The main verb and all its options are indicated in the section at the bottom of the graphical interface.

In the template shown in Figure 1, the example sentence is "I walk my dog", "I" is a variable of type personal pronoun, "walk" is the main verb, "my" is a variable of type possessive pronoun, "dog" is a variable of type animal and there is a dependency between "I" and "my" (which will allow to make their values agree in person, number and gender when generating all possible combinations).

In Figure 1 we also see that the user has selected the values "present and past" for the verb tense and "normal" and "imperative" for the verb form. Therefore, four sentences will be generated for each combination of the variables' values (one sentence for each combination of the tense and form selections). All these sentences will have the verb negated and will use the perfect tenses (as indicated by the extra verb options).

3 Testing the NLG Template Authoring Environment

In order to verify the correct functioning of the NLG Template Authoring Environment, we selected a set of sentence templates from the game "Business in Balance: Implementing an Environmental Management System" from DISTIL Interactive. The templates were selected manually, while keeping in mind the need to cover different aspects, as for example the number and type of the variables and dependencies. The testing of these examples covers for many more templates of the same type. The five selected sentence templates that form our testing set are displayed in Table 1 and are identified in the rest of this section by their reference number or order in the table.

Table 1. Testing examples

Ref. number	Template
1	`The ACTORS (ME/US) could help DEPARTMENTS.`
2	`The ACTORS IS/ARE now available to help.`
3	`I/WE struggled because of MY/OUR lack of knowledge.`
4	`I/WE AM/ARE pleased to report that I/WE completed the task TASKS.`
5	`I/WE WAS/WERE not the greatest choice for keeping things moving along quickly.`

In these template examples, we show in capitals the variable parts of the templates. `ACTORS`, `DEPARTMENTS` and `TASKS` refer to one of several possible nouns previously defined for each of the classes with those names. The terms in capitals separated by a "`/`" already display all the accepted values for that variable (for example `I/WE` represent a variable of type personal pronoun which could take only the selected values "I" or "we" and the rest are filtered out).

The first template example has two variables of predefined closed class nouns, `ACTORS` and `DEPARTMENTS`. The latter is independent, while the former has a dependency with a variable of type personal pronoun (in objective case form) that could only take the values "me" or "us". This template is used in the game when the actor/character available to help is the same actor/character that is providing the information. This template can be successfully generated with our system by declaring the variables, restricting the values of the pronoun variable, and establishing the dependency. When filtering non-valid sentences, the system will eliminate those cases where the value's number of the variable `ACTOR` and the personal pronoun do not agree (i.e., it will only allow sentences that use "me" if the actor is singular, and sentences that use "us", if the actor is plural). When creating this template, the user will have to be aware that the main verb is "to help" and indicate "could" as a modal to be used. This is important as otherwise SimpleNLG will modify the main verb in order to agree with the number of the subject. It is also necessary in case some of the options to change the main verb are specified.

Two examples of the generated sentences using the first template are shown below.

- `The HR Training Manager (me) could help the Design Department.`
- `The Implementation Team (us) could help the Deputy Management Representative.`

The second template is one that found a problem with our system and provided us with a reason and an opportunity to improve it. This example template also uses a variable of the closed class noun `ACTOR` together with the verb "to be" in the present tense, agreeing in number with the actor. It might seem trivial to indicate this dependency between the actor variable and the verb. But in our system the verbs are not treated as a regular variable (even when their values can be variable), but they are left for SimpleNLG to find the correct verb form. We needed then to inform SimpleNLG the number to which the verb should agree (by default it would assume singular). In this case we needed to inform SimpleNLG that the number to agree with would be the number of the variable `ACTOR`. We also have to consider the case when the subject number does not depend on a variable and is plural, as for example in a template where the subject is "`The members of DEPARTMENT`". To accommodate for these cases, we improved our system by asking the user to indicate in a pull down menu whether the template's verb should agree with a variable value or it should be always used in plural or in singular. (This option is displayed in the bottom right corner of the interface and not shown in the partial screen shot on Figure 1.)

The third template presents a dependency between a variable of type personal pronoun in the subjective case form, and a variable of type possessive pronoun in the complement. Both variables accept only a pair of their possible values, and the dependency between them establishes that they have to agree in person, number and gender. That is not a problem for our system. With respect to the verb, the user has to indicate the past tense as the only option.

In the fourth and fifth template, there is a personal pronoun variable taking the place of the subject, which should agree in person and number with the verb. This is, as mentioned before, left to SimpleNLG to solve. As the subject in these cases consists of only a personal pronoun and SimpleNLG can detect this fact, no extra information is required. In the fourth template, there is also a dependency between the personal pronoun variable in the subject role and the personal pronoun variable in the complement. Once again the person and number of these two variables have to agree, and the sentences not satisfying this restriction are filtered out by our system. Finally, for the fifth template the user is forced to specify that the verb "to be" has to be used in its past tense.

4 Conclusions and Future Work

We have identified the need for an NLG Template Authoring Environment that allows game content designers without linguistic and programming background to experiment with and finally create language templates.

We have designed and implemented a system that allows the user to specify an example sentence together with variables, its dependencies, and verb options that complete the template. This system shows the user all the possible sentences that could be generated with the specified template. It can be used to refine the template until it satisfies the user's needs.

The system makes use of the SimpleNLG java library which provides us with correct sentences and the possibility of including many verb variations, such as tense, form and modals.

We have evaluated our NLG Template Authoring Environment in a set of sentence templates from a digital-based interactive simulation game that covered different characteristics.

We have implemented a user-friendly intuitive graphical interface for the system. The convenience of use of this interface will be evaluated in the context of the development of a new game.

Acknowledgements

This work is supported by the Ontario Centres of Excellence (OCE) and Precarn Incorporated.

References

J. A. Bateman. 2002. Natural Language Generation: an introduction and open-ended review of the state of the art.

M. F. Caropreso, D. Inkpen, S. Khan and F. Keshtkar. 2009. Novice Friendly Natural Language Generation Template Authoring Environment. Proceeding of the Canadian Artificial Intelligence Conference 2009, Kelowna, BC, pp.195-198.

H. Dalianis. 1996. Concise Natural Language Generation from Formal Specifications, Ph.D. Thesis, (Teknologie Doktorsavhandling), Department of Computer and Systems Sciences, Royal Institute of Technology/ Stockholm University. Report Series No. 96-008, ISSN 1101-8526, SRN SU-KTH/DSV/R 96/8 SE.

K. van Deemter, E. Krahmer and M. Theune. 2005. Real versus Template-Based Natural Language Generation: A False Opposition? In Computational Linguistics, 31(1): 15-24.

D. French, C. Hale, C. Johnson and G. Farr. 1999. Internet Based Learning: An introduction and framework for higher education and business. London, UK: Kogan Page.

E. Reiter and R. Dale. 2000. Building Natural Language Generation Systems (Studies in Natural Language Processing), Cambridge University Press.

E. Reiter. 2007. SimpleNlg package: http://www.csd.abdn.ac.uk/ereiter/simplnlg