The Narrator: NLG for digital storytelling

Mariët Theune
Human Media Interaction
University of Twente
Enschede, The Netherlands
m.theune@utwente.nl

Nanda Slabbers
Human Media Interaction
University of Twente
Enschede, The Netherlands
nandaslabbers@hotmail.com

Feikje Hielkema*
Department of Computing Science
University of Aberdeen
Scotland, UK
fhielkem@csd.abdn.ac.uk

Abstract

We present the Narrator, an NLG component used for the generation of narratives in a digital storytelling system. We describe how the Narrator works and show some examples of generated stories.

1 Introduction

The automatic generation of narratives is still a largely unexplored field in NLG. Some exceptions are STORYBOOK (Callaway, 2000), a narrative prose generation system that can generate many different retellings of the same story (Little Red Riding Hood) and the architecture for a "narratologically enhanced NLG system" proposed by Lönneker (2005). An NLG system that is actually used in a digital storytelling application is PRINCE (Hervás et al., 2006).

Here we present the Narrator: the NLG component of the Virtual Storyteller, a multi-agent system that automatically creates fairy tales based on the actions of autonomous character agents in a simulated story world, where they can perform goal-oriented actions and experience emotions (Theune et al., 2004). The emerging story is captured in a formal representation and fed to the Narrator, which expresses it in natural language (in our case, Dutch). In the rest of this paper, we give a brief overview of the subsequent tasks the Narrator carries out to generate a fluent, well-formed narrative. We focus on the generation of referring expressions; a more detailed description of the entire generation process can be found in Theune et al. (2007). For more information concerning various design decisions see Theune et al. (2006).

* Feikje Hielkema carried out this work while she was at the University of Groningen, The Netherlands.

2 Document planning

The input for the Narrator is a Fabula (Swartjes and Theune, 2006): a story representation in the form of a causal network linking the following plot elements: actions, events, perceptions, goals, goal outcomes, and characters' "internal elements" such as emotions and beliefs. Possible links between these elements are motivation, enablement, mental and physical cause relations. Also, each plot element is associated with a time stamp (in terms of time steps in the story world). The (simplified) example Fabula in Figure 1 represents a very short story about a dwarf who is hungry and believes there is an apple in the house, leading to the goal to eat the apple. To achieve this, the dwarf carries out a simple plan: taking the apple and then eating it, which leads to the perception and the belief that the apple has been eaten, signifying a positive goal outcome.

As a first step in turning a Fabula into a Document Plan, all information that is not relevant for narration must be pruned away. An example is the standard perception-belief-positive outcome chain that follows a successful action: for the narrative, it is sufficient to mention only that the action was carried out. Currently, this process is not yet implemented in the Narrator; however, we assume that in the example Fabula, the nodes following the 'Eat Apple' action will be pruned away. The next step is to convert the pruned Fabula to a binary tree, replacing the causal links with appropriate rhetorical relations between plot elements. The basic rhetorical relations used in the Narrator are Cause, Contrast, Temporal and Additive, with more specific subclasses such as Purpose and Elaboration. When mapping the links in the Fabula to rhetorical relations, consecutive steps of a plan are connected using a Temporal

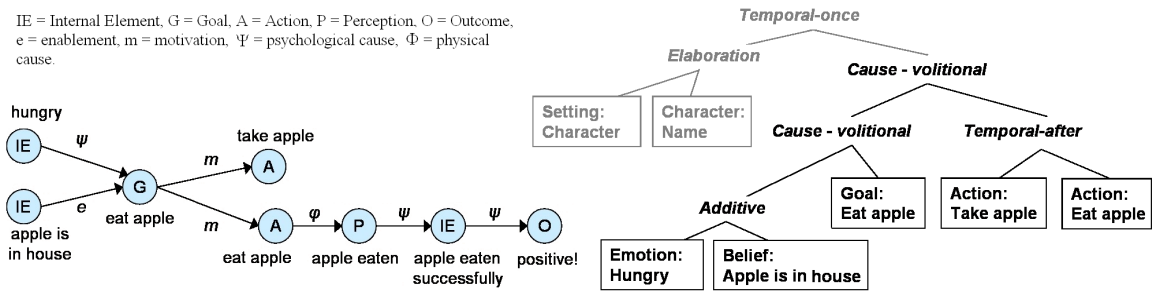

Figure 1: Fabula (left) and corresponding Document Plan (right).

relation. Motivation and psychological cause relations are mapped to Volitional Cause relations, and enablement and physical cause relations are mapped to Non-volitional Cause relations. Additive is the most general relation. It is used if two plot elements cause another plot element together, and more in general to connect two plot elements that do not have a more specific relation holding between them. We are currently investigating the automatic derivation of Contrast relations. The final step is to extend the Document Plan with a setting and background information about characters and objects. The example Document Plan in Figure 1 shows these extensions in grey: a Setting element introducing the protagonist, and an element specifying the name of the protagonist, connected via an Elaboration relation. They are in a Temporal-once (*Once upon a time...*) relation to the rest of the plot; this relation was added specifically for the fairy tale domain.

3 Sentence planning, lexicalisation and syntactic aggregation

Next, the leaves of the Document Plan are mapped to Dependency Trees. For each type of plot element a template is available telling exactly how its arguments should appear in the corresponding Dependency Tree. For example, actions are expressed using a straightforward active voice construction, with an optional PP argument to express instruments, e.g., *The knight opened the gate (with a key)*.[1] For internal states, there are templates for standard sentences such as *The princess was scared* but also for storytelling-style constructions such as *Oh, how happy she was!* and *She had never been so happy!*, to be used for emotions with a high intensity. After the Dependency Trees are selected, their nodes are mapped to Dutch words (except the nodes referring to entities, which are lexicalised as part of

[1]For reading ease, Narrator output is translated to English.

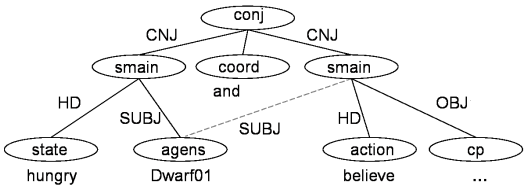

Figure 2: Aggregated Dependency Tree.

referring expression generation). The lexical choice algorithm uses a discourse history, to achieve some variation in wording by taking into account which words have been used recently. The words added to the Dependency Trees are still uninflected, as morphology is taken care of during Surface Realization.

The Narrator uses a syntactic aggregation algorithm that combines pairs of Dependency Trees and adds an appropriate cue phrase to signal their rhetorical relation. The properties of this cue phrase determine which syntactic construction is used to combine the Dependency Trees. If the resulting tree contains repeated elements, these can be ellipted. Figure 2 shows an example where the subject of the second clause is deleted (Conjunction Reduction). A corresponding surface string could be *The dwarf was hungry and believed there was an apple in the house*, expressing the Additive relation in the Document Plan of Figure 1. To keep the aggregated sentences from getting too complex, at most three simple Dependency Trees can be combined. In cases where this restriction prohibits aggregation, rhetorical relations are expressed using adverbs such as *then*, *however* etc. For a more detailed description of the aggregation process, see Theune et al. (2006).

4 The generation of referring expressions

To determine whether a pronoun or a noun should be used to refer to a certain entity, we use a variant of the algorithms of McCoy and Strube (1999) and Henschel et al. (2000). Based on an analysis of

human-written fairy tales we determined that pronouns are dispreferred (1) at the beginning of a paragraph, (2) if the antecedent has not been mentioned for two sentences, (3) if a pronoun has been used four times and the referring expression is the first one in the sentence. Also, pronouns cannot be used when the referring expression should express additional information (e.g., about a character's emotional state) in the form of an adjective or a relative clause. If the above conditions do not hold, a pronoun is used if there is either strong parallelism or a causal relationship with the previous clause or sentence (Chambers and Smyth, 1998; Kehler, 2002); otherwise the decision is based on the salience of the referent (Lappin and Leass, 1994).

If a noun phrase is to be generated, the referent's name (if available) is used in 25% of the cases, either on its own or in a construction such as *princess Amalia* if the noun describes a function, such as princess, king or knight. If a description is to be generated, first a noun has to be selected. Some concepts have both a preferred lexicon entry (the most common word for that concept) and one or more additional entries that are used occasionally for variation. After having selected the noun, different types of adjectives can be added. Distinguishing adjectives, necessary to create an unambiguous referring expression, are selected only for subsequent references using a modified version of the algorithm of Krahmer and Theune (2002). When introducing a new entity all its properties are mentioned, because they can be used as distinguishing properties later in the story. Non-distinguishing adjectives include adjectives describing a character's internal state, and 'decorational' adjectives used to spice up the descriptions of entities that have no specific properties except their type (e.g., *a heavy gate*). Finally, an indefinite article is added when the entity has not been mentioned before, and a definite article otherwise.

Simple inference rules are used to generate bridging descriptions. If the referent (e.g., a gate) is related to a discourse-old entity (e.g., a castle), the algorithm checks if there is a rule saying "all castles have gates". If there are no other salient entities for which the same rule holds (i.e., that they always have gates), then a bridging description like *the gate* can be used instead of *the gate of the castle*.

5 Surface form generation

When referring expression generation is finished, the Surface Realiser linearises the now fully lexicalised Dependency Trees. It traverses them depth-first, ordering the children of each node by grammar rules such as SMAIN → SU + HD + OBJ, which states that if a parent node has syntactic category 'SMAIN' (sentence) and three children with dependency labels 'SU' (subject), 'OBJ' (direct object) and 'HD' (main verb), then those children should be ordered in the above way. This particular rule would for instance be applied to produce the sentence *The prince loved Amalia*. Any nouns, adjectives and verbs are inflected at the moment they are linearised. Punctuation is added once linearisation is complete. This concludes our description of the Narrator; for more details see (Theune et al., 2007).

6 Examples

Our simple example story about the hungry dwarf is narrated as follows:

Once upon a time there was a dwarf called Plop. He was hungry and believed there was an apple in a house.[2] *Therefore he wanted to eat the apple. After Plop had taken the apple, he ate it.*

This story is well-formed and coherent, but also quite simple. The next story better illustrates the Narrator's potential, including examples of Document Plan extension, aggregation, pronominalization, and the use of 'decorational' adjectives (*a high tree*). It was generated from a hand-made Document Plan (shown in Figure 3), which contains Contrast relations and paragraph boundaries that cannot currently be generated automatically by the Document Planner. So the story illustrates the output quality that could be achieved by the Narrator once these remaining Document Planning problems are resolved:

Once upon a time there was a beautiful princess called Amalia. A knight from a far away country was in love with her, but she was in love with a young prince. The knight was jealous, so he wanted to abduct her.

After the knight had climbed a high tree, he jumped into the princess' bedroom. She was so scared that she screamed loudly, but nobody heard her.

The knight grabbed the princess and then he placed her on his horse. After that he took her to an old and narrow bridge. On the other side she saw the prince she was in love with. Oh, how relieved princess Amalia was!

[2] Assuming that the house belongs to Plop, the bridging description *the house* would be more appropriate here. However, in this case the Narrator lacked knowledge about the owner of the house, so a general indefinite description was produced.

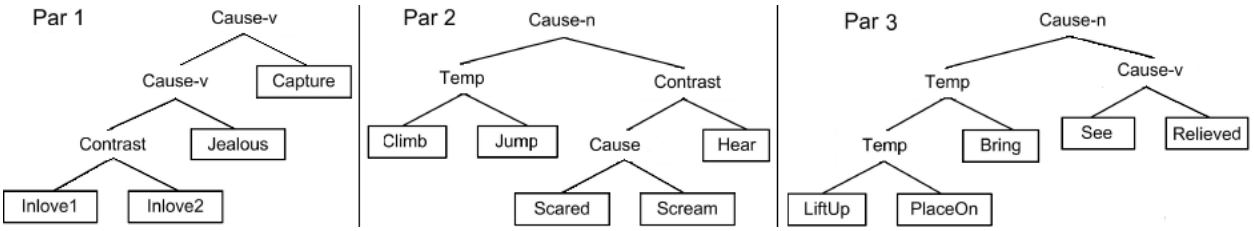

Figure 3: Initial Document Plan for the second example story.

7 Concluding remarks

The Narrator has been implemented (in Java), but so far has only been tested with hand-made input structures, because parts of the Document Planner and of the Virtual Storyteller's plot generation component are still under construction. So far, the only evaluations have been informal comparisons with earlier versions. The Narrator does not employ the kind of narratological knowledge proposed by Lönneker (2005), and unlike STORYBOOK it cannot handle narrative aspects such as multiple viewpoints or character dialogue. However, it can generate well-formed and fluent stories containing some typical narrative constructions. Currently being investigated are the automatic placement of paragraph boundaries, detection of contrast relations and lexicalisation of emotions, taking their intensity into account. In addition, as pointed out by one of our reviewers, it would be beneficial to add a form of semantic aggregation to the system, grouping related plot elements together during document planning.

Our main long-term challenge is to generate texts that are not only grammatical and coherent, but that can also really affect the reader by employing narrative techniques such as the use of subjective perspectives to heighten identification, and foreshadowing to increase suspense. Ablation tests in the style of Callaway (2000) could then be used to evaluate the effect of such techniques.

References

C. Callaway. 2000. *Narrative Prose Generation*. Ph.D. thesis, North Carolina State University, Raleigh, NC.

G.C. Chambers and R. Smyth. 1998. Structural parallelism and discourse coherence: A test of Centering Theory. *Journal of Memory and Language*, 39:593–608.

R. Henschel, H. Cheng, and M. Poesio. 2000. Pronominalization revisited. In *Proceedings of COLING*, pages 306–312.

R. Hervás, F. Pereira, P. Gervás, and A. Cardoso. 2006. Cross-domain analogy in automated text generation. In *Proceedings of the Third joint workshop on Computational Creativity, ECAI'06*.

A. Kehler. 2002. *Coherence, Reference, and the Theory of Grammar*. CSLI Publications.

E. Krahmer and M. Theune. 2002. Efficient context-sensitive generation of referring expressions. In K. van Deemter and R. Kibble, editors, *Information Sharing: Reference and Presupposition in Language Generation and Interpretation*, pages 223–264. CSLI Publications.

S. Lappin and H. Leass. 1994. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535–561.

B. Lönneker. 2005. Narratological knowledge for natural language generation. In *Proceedings of the 10th European Workshop on Natural Language Generation (ENLG-05)*, pages 91–100.

K.E. McCoy and M. Strube. 1999. Generating anaphoric expressions: Pronoun or definite description? In *Proceedings of the ACL Workshop on The Relation of Discourse/Dialogue Structure and Reference*, pages 63–71.

I. Swartjes and M. Theune. 2006. A Fabula model for emergent narrative. In *Technologies for Interactive Digital Storytelling and Entertainment (TIDSE)*, LNCS 4326, pages 95–100.

M. Theune, S. Rensen, R. op den Akker, D. Heylen, and A. Nijholt. 2004. Emotional characters for automatic plot creation. In *Technologies for Interactive Digital Storytelling and Entertainment (TIDSE)*, LNCS 3105, pages 95–100.

M. Theune, F. Hielkema, and P. Hendriks. 2006. Performing aggregation and ellipsis using discourse structures. *Research on Language and Computation*, 4(4):353–375.

M. Theune, N. Slabbers, and F. Hielkema. 2007. The automatic generation of narratives. In *Proceedings of CLIN-17*. to appear.