

Generation Models for Spoken Dialogues

Graham Wilcock and Kristiina Jokinen

University of Helsinki
00014 Helsinki, Finland

graham.wilcock@helsinki.fi
kristiina.jokinen@helsinki.fi

Abstract

The paper discusses what kind of generation model is suitable for spoken dialogue responses. We describe different existing models of generation, and compare them from the point of view of spoken dialogue systems. We introduce a model of generation based on new information focus, and argue that this addresses the communicative requirements of flexible spoken dialogue systems (incrementality, immediacy and interactivity). We discuss the relationship between the dialogue manager and the generator, and what kind of interface they should share. We describe a flexible shallow generation approach which combines template-based generation with a pipeline of distinct processing levels.

1 Introduction

In this paper we address the question of what kind of generation is suitable for spoken dialogue systems. We look at the question from three perspectives.

First, in Section 2 we discuss different kinds of generation models which are in use in different fields (machine translation, text generation, telephone booking systems) and consider whether they are appropriate for spoken dialogue systems. We argue that for spoken dialogue systems a model of generation must take account of new and old information status, must allow for interactivity with the user, and must be adaptive to different user expectations.

Second, in Section 3 we consider the relationship between the dialogue manager and the generator, and what kind of interface they should share. We argue that for spoken dialogue systems an agenda type of interface is

suitable, in which new and old information status is already marked up by the dialogue manager to simplify the work of the generator.

Third, in Section 4 we discuss the question of whether template-based generation methods are satisfactory for dialogue responses. We argue that in the NLG community the question of template-based versus plan-based NLG is now seen as a false opposition, and the real issue is how to make templates more flexible and how to use them in practical “shallow generation” approaches.

The paper is based on practical experience in designing and implementing spoken dialogue systems, most recently the USIX-Interact project (Jokinen et al., 2002). We will be glad to offer a demo showing our model of response generation, using an XML-based NLG implementation and a Java speech synthesizer.

2 Models of Generation

In this section we describe different models of generation, originating from work in machine translation, text generation, and telephone booking systems. We consider whether they are appropriate for spoken dialogue response generation, and argue that a model of generation based on new information focus is more suitable.

2.1 A Machine Translation Model

Machine translation systems concentrate mainly on analysis and transfer problems, so the generation stage is often rather simple. Some MT systems which take generation more seriously use bidirectional grammars. In this case generation may be performed by a bag generation algorithm, in which generation starts from a bag of semantic terms or lexical

items. An essential property of the generation algorithm is that the order of information in the bag is not significant - it has no effect on surface realization order.

In machine translation, treating the source information as unordered is attractive, as the surface order in the target language should not be tied to the surface order in the source language. The usual approach in bag generation has therefore been to generate all possible grammatical sentences (all sentences licensed by the target language grammar) which match the source information (i.e. use all and only the items in the source bag). This produces a combinatorial explosion of different sentences which are grammatically legal. The emphasis of research efforts in bag generation has therefore been on finding ways to increase the efficiency of the algorithms (Carroll et al., 1999).

In this model, the generation problem has been seen either as purely syntactic (find all syntactically permitted sentences which use the given lexical items), or as a relationship between semantics and syntax (find all syntactically permitted sentences which realize the given semantic terms). Usually, information structure is ignored. This model does not attempt to select the best realization from the multiple possible alternatives, if they are all syntactically equally correct.

A similar kind of approach is used in Nitrogen (Langkilde and Knight, 1998) where a symbolic generator is combined with statistical selector, and sentence generation is basically based on statistical models of sentence structure derived from large corpora.

The form of bag generation developed for machine translation has also been used for dialogue response generation. The algorithm is suitable for incremental generation, and an implementation of an incremental version of the algorithm is described by Wilcock (1998). In this incremental model, the initial bag of terms is incomplete, and more semantic terms are added after generation has started. The generator starts generating from the incomplete bag, and must find a way to include the additional semantic terms when they are supplied. In this model, although the order of terms inside the bag is not significant, the order in which

terms are added influences the utterance very strongly.

Wilcock (1998) suggests *utterance rules* to control the behaviour of the generator. From an incomplete bag and a partial utterance, the generator attempts to continue the utterance as further semantic terms are added. When the generator cannot find a way to continue, a repair is performed. Like the underlying bag model, however, this incremental model does not take information structure explicitly into account as a significant factor in generation.

2.2 A Text Generation Model

In the model of generation adopted in text generation systems (Reiter and Dale, 2000), information structure is recognised as a major factor. This model usually has a pipeline architecture, in which one stage explicitly deals with discourse planning and another stage deals with referring expressions, ensuring that topic shifts and old and new information status are properly handled.

Although the text generation model is organized to deal effectively with new and old information structure, it is designed for producing text which is basically monologue rather than dialogue. Spoken dialogue systems impose different requirements on generation, such as how to handle prosody, which do not arise in text generation systems.

A basic point about the monologue nature of text generation is that it is essentially a one-shot process. All of the information to be expressed must be put into the generated text, so that it is available for the reader to extract later. In a dialogue this is not necessary. Part of the relevant information can be given initially, and the rest of the information can be given later depending on the user's reactions to the first part. The user may in fact indicate that the rest of the information is not wanted.

In spoken interaction, listeners also immediately react to the information units carried by the utterance, and the speaker can modify the following utterances according to this kind of feedback. Hence, faced with one of the fundamental questions of NLG namely "where does generation start from?", which brings the danger of sliding down a slippery slope (McDon-

ald, 1993), we can reply that in dialogue systems generation starts simultaneously with interpretation, as a reaction to the presented information. The initial 'message' is then gradually specified into a linguistic expression with respect to language-specific knowledge and communicative requirements of the situation.

In a dialogue system, as Santamarta (1999) says, "the length of the generated text is rather short, namely a turn. A turn can consist of one word up to a couple of sentences." The text planning part of the text generation model, which plans how to pack all of the information into a single text, is therefore unsuitable as a model for dialogue response generation where content determination is handled by the dialogue manager and text planning is rather simple.

Nevertheless we advocate using a version of the pipeline architecture which is common in text generation systems. The different stages of the pipeline perform specialized subtasks, and many of these tasks (including lexicalization, referring expressions, aggregation) are equally required in generation for a spoken dialogue system.

2.3 A Telephone Bookings Model

Another model of generation, developed for telephone-based booking and ordering systems, is based on the use of dialogue grammars or dialogue description languages, such as VoiceXML (VoiceXML Forum, 2000). Unlike the text generation model, this approach is explicitly designed for dialogue systems rather than monologue systems. This model, however, suffers from the disadvantage that it tends to increase the rigidity of the system, by enforcing a form-filling approach which makes the user fit in with the system's demands. The generation of system utterances is mainly a matter of designing prompts, i.e. the system responses are carefully tailored so as to elicit appropriate type of reaction from the user.

This model fits in the common view of human-computer interaction where the computer is regarded as a tool which supports human goals: the role of the computer is to be a passive and transparent "slave" under human control. However, in recent years, another

metaphor has also become available: the computer as an agent which is capable of mediating interaction between human users and the application (cf. the situated delegation-oriented dialog paradigm in SmartKom, (Wahlster et al., 2001)). Advanced research and technological development allows us to build more intelligent software which takes into account requirements for complex interaction: spoken dialogue systems contain sophisticated models of communication and of the user, and the interest in multimodality also extends the system's communicative repertoire. Furthermore, information-providing systems often encounter situations where they need to present large amounts of complex information to the user and they need to present this in a form that is accessible and clear. Consequently, dialogue systems should also have more sophisticated models of generation in order to be able to present complex information to the user.

3 Dialogue Manager and Generator

In this section we describe the model of generation which we advocate for spoken dialogue systems, and this leads us to consider the relationship between the dialogue manager and the generator, and what kind of interface they should share. We argue that for spoken dialogue systems an agenda type of interface is suitable, in which new information status is already marked up by the dialogue manager.

3.1 A NewInfo-based Model

The model of generation which we advocate and which we use in our spoken dialogue systems is described by Jokinen et al. (1998). This model of generation, which focusses on incrementality, immediacy and interactivity (the three I's), is more suitable for highly interactive systems than the approaches discussed previously.

In this model, response planning starts from the new information focus, called *NewInfo*. The generator decides how to present *NewInfo* to the user: whether to present it by itself or whether to wrap it in appropriate linking information. The following examples are taken

from Jokinen and Wilcock (2001), where they are discussed in more detail.

- (1) User: *Which bus goes to Malmi?*
System: *Number 74.*
- (2) User: *How do I get to Malmi?*
System: *By bus - number 74.*

In example (1) NewInfo is the information about the bus number, while in (2) NewInfo concerns the means of transportation. In both cases, NewInfo is presented to the user by itself, without linking to the Topic.

- (3) *When will the next bus leave for Malmi?*
 - (a) *2.20pm.*
 - (b) *It will leave at 2.20pm.*
 - (c) *The next bus to Malmi leaves at 2.20pm.*

Whether NewInfo should be wrapped or not depends on the changing dialogue context. When the context permits a fluent exchange of contributions, wrapping is avoided and the response is based on NewInfo only, as in example (3a). When the context requires more clarity and explicitness, NewInfo is wrapped by Topic information as in (3b) in order to avoid misunderstanding. When the communication channel is working well, wrapping can be reduced, but when there are uncertainties about what was actually said, wrapping must be increased as in example (3c) to provide implicit confirmation.

3.2 The Agenda

In this approach, the dialogue manager creates an *Agenda*, a set of domain concepts which it makes available for use by the generator. The generator can freely use the concepts in the agenda in order to realise the system's intention as a surface string, but it is not forced to include in the response all the concepts that the dialogue manager has designated as relevant in the agenda.

Because the dialogue manager is responsible for recording dialogue history, performing topic tracking, and getting new information from the data source (via a task manager), it is the best authority to decide the new or old information status of each concept. The dialogue

manager therefore marks up each concept in the agenda with Topic and NewInfo tags. This information is very useful to the generator in deciding how to realize the concepts in the generated response.

Thus the dialogue manager and the generator communicate via the specifically marked conceptual items in the shared agenda, but they both make their own decisions on the basis of their own reasoning and task management. The dialogue manager need not know about particular rules of surface realisation while the generator need not know how to decide the information status of the concepts in the current dialogue situation.

Here is the agenda given by Jokinen and Wilcock (2001) for response (2) *By bus - number 74*. All information in the system is held in XML format, including the agenda. Note that this is a simplified early prototype format - later agenda formats use Annotation Graph (Bird et al., 2001) representations.

```
<agenda>
  <concept info="NewInfo">
    <type>means-of-transportation</type>
    <value>bus</value>
  </concept>
  <concept info="Topic">
    <type>destination</type>
    <value>malmi</value>
  </concept>
  <concept info="NewInfo">
    <type>bus</type>
    <value>exists</value>
  </concept>
  <concept info="NewInfo">
    <type>busnumber</type>
    <value>74</value>
  </concept>
</agenda>
```

If the real-time requirements of the system allow sufficient time, the generator can decide on the optimum way to wrap the new information, but if there is extreme urgency to produce a response, the generator can simply give the new information without wrapping it. If this leads to misunderstanding, the system can attempt to repair this in subsequent exchanges. In this sense, the model offers an *any-time* algorithm.

4 Template-based Generation

In this section we discuss the question of whether template-based generation methods are satisfactory for dialogue response generation.

The role of template-based generation has been widely debated in the NLG community (Becker and Busemann, 1999), and the question of template-based versus plan-based NLG is now largely seen as a false opposition. The real issue is how to make templates more flexible and how to use them in practical “shallow generation” approaches.

The problem is that “deep” NLG needs to perform linguistic processes at a number of different levels (lexicalization, referring expressions, aggregation) whereas templates have traditionally been simple, single-shot, single-level mechanisms. The simplest kinds of templates use only canned texts and are single-level, but more flexible templates can include a range of information at different linguistic levels.

Following van Deemter et al. (1999), we take *template-based* to mean “making extensive use of a mapping between semantic structures and representations of linguistic surface structures that contain gaps”. On this interpretation, templates can clearly include linguistic knowledge and the ways in which the gaps can be filled can be highly flexible.

4.1 Shallow Generation

Our approach is to use templates to create initial tree structures which serve as simple text plans for short dialogue responses. We then pass these tree structures through a pipeline of processes which perform specialized operations at a number of different linguistic levels, adding, deleting or modifying the information in the structure. This pipeline is based on the standard text generation pipeline architecture described by Reiter and Dale (2000).

However, we advocate a form of “shallow generation” as described by Busemann and Horacek (1998), which allows the pipeline components to be simpler and more lightweight than in a full-scale text generation system. The shallow generation approach does not attempt general solutions and does not try to model everything. It restricts itself to domain-specific and

task-specific choices, with a linguistic coverage based only on a domain corpus.

If the processing of the initial template-based tree structure by the pipeline stages is so extensive that the original template vanishes completely, it would be inappropriate to call this approach “template-based”, but in our shallow generation system the template’s structure is retained through most of the levels. The processing levels modify the information inside the template, for example replacing the domain concept nodes in the tree structure with linguistic referring expression nodes.

4.2 A Pipeline Architecture

The prototype response generator described by Jokinen and Wilcock (2001) has a simple pipeline consisting of an aggregation stage, a combined lexicalization and referring expressions stage, and a surface realization stage.

Following the NewInfo-based model, the generator selects from the agenda those concepts marked as NewInfo as the basis for generation, and also decides whether NewInfo will be the only output, or whether it will be preceded by Topic linking concepts.

The aggregation stage decides on an appropriate template, based on the concepts in the agenda and their status as Topic or NewInfo. In the aggregation stage, variable slots in the templates are filled simply by copying concepts from the agenda into the slots, pending further processing by the lexicalization and referring expression stages.

In the lexicalization and referring expression stages, concepts in the aggregation templates are replaced by lexical items and referring expressions using XML-to-XML transformations. Further details of the XML-based implementation are discussed by Wilcock (2001).

The final stages of the pipeline perform syntactic and morphological realization. The early prototype generates simple responses like the examples in Section 3.1. Later versions of the system can generate more complex responses in both Finnish and English.

4.3 NLG and Speech

The output from the NLG pipeline is marked up for speech synthesis in Java Speech Markup

Language (Sun Microsystems, 1999). The response *By bus - number 74* in example (2) is marked up in JSML as follows.

```
<jhtml>
  <div type="sent"> By
    <emphasis> bus </emphasis>
  </div>
  <break size="large"/>
  <div type="sent"> Number
    <sayas class="number"> 74 </sayas>
  </div>
</jhtml>
```

Here `<div type="sent">` marks sentence boundaries, `<emphasis>` shows that the word *bus* should be spoken with emphasis, `<break>` shows that a pause is required before the second part of the response, and `<sayas class="number">` tells the speech synthesizer that 74 should be pronounced “seventy-four”.

In our implemented system, speech synthesis is performed by the open source FreeTTS Java speech synthesizer (Sun Microsystems, 2002).

5 Conclusion

We have discussed the design of a generation component for a spoken dialogue system, and argued that the flexibility needed in spoken dialogue systems can be addressed by a suitable generation model. Our model of NewInfo-based generation supports incrementality, immediacy and interactivity. The design combines template-based generation with a pipeline architecture within a shallow generation approach, using an XML-based implementation.

A working system can be demonstrated at the workshop.

References

- Tilman Becker and Stephan Busemann, editors. 1999. *May I Speak Freely? Between Templates and Free Choice in Natural Language Generation*. Proceedings of the KI-99 Workshop. DFKI, Saarbrücken.
- Steven Bird, Kazuaki Maeda, and Xiaoyi Ma. 2001. AGTK: The annotation graph toolkit. In *IRCS Workshop on Linguistic Databases*, Philadelphia.
- Stephan Busemann and Helmut Horacek. 1998. A flexible shallow approach to text generation. In *Proceedings of the Ninth International Workshop on Natural Language Generation*, pages 238–247, Niagara-on-the-Lake, Ontario.
- John Carroll, Anne Copestake, Dan Flickinger, and Victor Poznanski. 1999. An efficient chart generator for (semi-)lexicalist grammars. In *7th European Workshop on Natural Language Generation*.
- Kristiina Jokinen and Graham Wilcock. 2001. Confidence-based adaptivity in response generation for a spoken dialogue system. In *Proceedings of the 2nd SIGdial Workshop on Discourse and Dialogue*, pages 80–89, Aalborg, Denmark.
- Kristiina Jokinen, Hideki Tanaka, and Akio Yokoo. 1998. Planning dialogue contributions with new information. In *Proceedings of the Ninth International Workshop on Natural Language Generation*, pages 158–167, Niagara-on-the-Lake, Ontario.
- Kristiina Jokinen, Antti Kerminen, Mauri Kaipainen, Tommi Jauhiainen, Graham Wilcock, Markku Turunen, Jaakko Hakulinen, Jukka Kuusisto, and Krista Lagus. 2002. Adaptive dialogue systems - Interaction with Interact. In *Proceedings of the 3rd SIGdial Workshop on Discourse and Dialogue*, pages 64–73, Philadelphia.
- Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL'98)*, pages 704–710.
- David McDonald. 1993. Does natural language generation start from a specification? In H. Horacek and M. Zock, editors, *New Concepts in Natural Language Generation*, pages 275–278. Pinter, London.
- Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press.
- Lena Santamarta. 1999. Output generation in a spoken dialogue system. In Becker and Busemann (1999), pages 58–63.

- Sun Microsystems. 1999. Java Speech Markup Language Specification, version 0.6. <http://java.sun.com/products/java-media/speech/forDevelopers/JSML>.
- Sun Microsystems. 2002. FreeTTS: A speech synthesizer written entirely in the Java programming language. <http://freetts.sourceforge.net/>.
- Kees van Deemter, Emiel Krahmer, and Mariët Theune. 1999. Plan-based vs. template-based NLG: A false opposition? In Becker and Busemann (1999), pages 1–5.
- VoiceXML Forum. 2000. Voice eXtensible Markup Language VoiceXML, Version 1.00. <http://www.voicexml.org/>.
- Wolfgang Wahlster, Norbert Reithinger, and Anselm Blocher. 2001. SmartKom: Multimodal communication with a life-like character. In *Proceedings of Eurospeech 2001*, Aalborg, Denmark.
- Graham Wilcock. 1998. Approaches to surface realization with HPSG. In *Proceedings of the Ninth International Workshop on Natural Language Generation*, pages 218–227, Niagara-on-the-Lake, Ontario.
- Graham Wilcock. 2001. Pipelines, templates and transformations: XML for natural language generation. In *Proceedings of the 1st NLP and XML Workshop*, pages 1–8, Tokyo.