Computer Speech and Language (2002) **16**, 387–407
doi: 10.1016/S0885-2308(02)00012-8
Available online at http://www.idealibrary.com on IDEAL®

Stochastic natural language generation for spoken dialog systems

Alice H. Oh*† and Alexander I. Rudnicky

Carnegie Mellon University, School of Computer Science, 5000 Forbes Avenue, Pittsburgh, PA 15213 U.S.A

Abstract

We describe a corpus-based approach to natural language generation (NLG). The approach has been implemented as a component of a spoken dialog system and a series of evaluations were carried out. Our system uses n-gram language models, which have been found useful in other language technology applications, in a generative mode. It is not yet clear whether the simple n-grams can adequately model human language generation in general, but we show that we can successfully apply this ubiquitous modeling technique to the task of natural language generation for spoken dialog systems. In this paper, we discuss applying corpus-based stochastic language generation at two levels: content selection and sentence planning/realization. At the content selection level, output utterances are modeled by bigrams, and the appropriate attributes are chosen using bigram statistics. In sentence planning and realization, corpus utterances are modeled by n-grams of varying length, and new utterances are generated stochastically. Through this work, we show that a simple statistical model alone can generate appropriate language for a spoken dialog system. The results describe a promising avenue for using a statistical approach in future NLG systems.

1. Introduction

Natural language generation is the process of generating text from a meaning representation. It can be thought of as the reverse of natural language understanding (NLU, see Fig. 1). While it is clear that NLG is an important part of natural language processing, there has been considerably less research activity in NLG than in NLU. This can be partly explained by the reality that NLU, at least until now, has found a larger number of practical applications, for example, in the processing of potentially large amounts of text and speech, and as part of natural-language interfaces. By contrast, NLG systems provide output for other systems, from automatically created representations.

* Present address: MIT Artificial Intelligence Laboratory, 200 Technology Sq. Rm. 812, Cambridge, MA 02139, U.S.A.
† *E-mail address:* aoh@ai.mit.edu (A.H. Oh).

0885-2308/02/$35

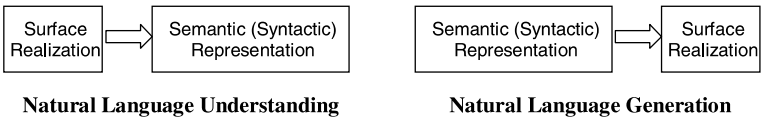

Figure 1. NLU and NLG.

Nevertheless, NLG plays a critical role in applications such as text summarization, machine translation, and dialog systems. Presently, the focus of research has been primarily on text generation, with correspondingly less attention paid to the development of adequate NLG technologies for spoken dialog systems, despite the importance of good NLG in such systems. The present work, along with other related efforts, such as Baptist and Seneff (2000) and Ratnaparkhi (2000), focuses on just this problem.

In the process of developing and maintaining a natural language generation (NLG) module for a spoken dialog system, we recognized the limitations of the current NLG technologies for our purposes. While several general-purpose rule-based generation systems have been developed (cf. Elhadad & Robin, 1996), because of their generality they are often quite difficult to adapt to small, task-oriented applications. Several different solutions have been proposed to overcome this shortcoming. For example, Bateman and Henschel (1999) have described a lower cost and more efficient generation system for a specific application using an automatically customized subgrammar. Busemann and Horacek (1998) describe a system that mixes templates and rule-based generation. This approach takes advantages of templates and rule-based generation as needed by specific sentences or utterances. Stent (1999) has also proposed a similar approach for a spoken dialog system. However, each one of these approaches still imposes the requirement of writing grammar rules and acquiring the appropriate lexicon, a specialist activity.

Because comparatively less effort and linguistic expertise is needed, many current dialog systems use template-based generation. But there is one obvious disadvantage to templates: the quality and appropriateness of the output depends entirely on the set of templates. Even in a relatively simple domain, as travel reservations, the number of templates necessary for reasonable quality can become quite large to the extent that maintenance becomes a serious problem. There is an unavoidable tradeoff between the amount of time and effort in creating and maintaining templates and the variety and quality of the output utterances. This will become clear when we present the details of the template system we developed, in Section 4.1.

In response to the limitations of rule-based and template-based techniques, we developed a novel approach to natural language generation. It features a simple, yet effective, corpus-based technique that uses statistical models of task-oriented language spoken by domain experts to generate system utterances. We have applied distinct n-gram-based techniques to sentence realization and to content planning, and have incorporated the resulting generation component into a working spoken dialog system.

To evaluate our new generation module, we ran comparative evaluations on our spoken dialog system using different NLG components. We found that our new technique performs well for our spoken dialog system (i.e., users liked the output), and thereby demonstrated that the corpus-based NLG is a promising direction for further exploration. In addition, we gained some insight into the effectiveness of usability-based evaluation methods for NLG modules embedded within a larger system.

2. Natural language generation for spoken dialog systems

Since current NLG systems have been mostly developed for text-based applications, applying existing NLG techniques directly to spoken dialog systems poses some problems. Recently, research has begun that looks at how to design effective NLG modules for spoken dialog systems (see for example, Baptist & Seneff, 2000; Ratnaparkhi, 2000; Walker *et al.*, 2001).

A spoken dialog system enables human–computer interaction via spoken natural language. A task-oriented spoken dialog system speaks as well as understands natural language as part of the process of completing a well-defined task. Examples of research systems include a complex travel planning system (Rudnicky *et al.*, 1999), a publicly available worldwide weather information system (Zue, 2000), and an automatic call routing system (Gorin, Riccardi, & Wright, 1997).

Building an NLG component for a spoken dialog system differs from more traditional applications of NLG, such as generating documents, and provides a novel way of looking at the generation problem. The following are some characteristics of task-oriented spoken dialog systems that define unique challenges for spoken dialog NLG.

1. The language used in spoken dialog is different from the language used in written text. Spoken utterances are generally shorter in length compared to sentences in written text. Spoken utterances follow grammatical rules, but much less strictly than written text. Also, the syntactic structures used tend to be much simpler and less varied than those in written text. In a spoken dialog, simple and short utterances may be easier to say and understand, since these impose a smaller cognitive load on the listener. More effective communication is thereby possible.
2. The language used in task-oriented dialogs tends to be very domain-specific. The domains for these systems are fairly narrow. This means that the lexicon for a given spoken dialog system can be fairly small and domain-specific.
3. NLG is often not the main focus in building or maintaining dialog systems. Yet the NLG module is critical in development and system performance: In a telephone-based dialog system, NLG and text-to-speech (TTS) synthesis are the only components that users will experience directly. But with limited development resources, NLG has traditionally been overlooked by spoken dialog system developers.

Taking these characteristics into account, NLG for task-oriented spoken dialog systems must be able to

1. generate language appropriate for spoken interaction; system utterances must be easily comprehended, that is, not be too lengthy or be too complex (e.g., syntactically);
2. generate domain-specific language; the lexicon must contain appropriate words and expressions for the domain;
3. enable fast prototyping and development of the NLG module. A serviceable generation capability needs to be available fairly early in the development cycle.

Also, to forward the overall goals of the spoken dialog system, the NLG component must do its part in supporting a natural conversation. That is, the NLG output must elicit appropriate responses from the user, prevent user confusion, and guide the user in cases of confusion. We have observed that while the NLG component is not wholly responsible for creating these desirable characteristics of a spoken dialog system, it certainly can contribute (as seen in commercial dialog

systems where system prompts are designed with great care to resolve these issues, see Kotelli, 2001).

3. Existing approaches

Many research NLG systems use generation grammar rules, much like parsers with semantic or syntactic grammars. A good example of a rule-based system is SURGE (Elhadad & Robin, 1996). In general, well-written generation grammar rules enable an NLG system to have wide coverage, be domain independent, and be reusable, proven by many very different applications that use SURGE.

However, input to SURGE, and other rule-based systems in general, needs to be richly specified with features such as verb tense, number, NP heads, categorization, definiteness, and others (see Fig. 2). This is one of the greatest disadvantages of the rule-based technique. For a rule-based system to be more efficient than a template-system, the amount of time and effort that goes into designing and adhering to such a rich input representation must be justified. This is probably why most spoken dialog systems use templates rather than grammar rules. Most rule-based systems do provide default values, but using the default values too often defeats the purpose of rule-based technique; it would produce output equivalent to a poorly developed set of templates. With richly specified input, though, SURGE can produce a wide variety of sentences.

Also, as is true for parsing grammars, generation grammar rules take much effort and time to develop. The set of grammar rules is the core knowledge source of any rule-based NLG systems, and the quality of output and coverage of sentence types depend on the set of grammar rules. Not only do they take time and effort, only a highly skilled grammar writer can write the rules. One advantage, though, is that most rules are domain independent, so once developed, they can be reused in many applications, provided the input specification conforms.

Templates provide an alternative approach, in which the developer handcrafts a set of templates and canned expressions. As evidenced by the frequent use of templates in

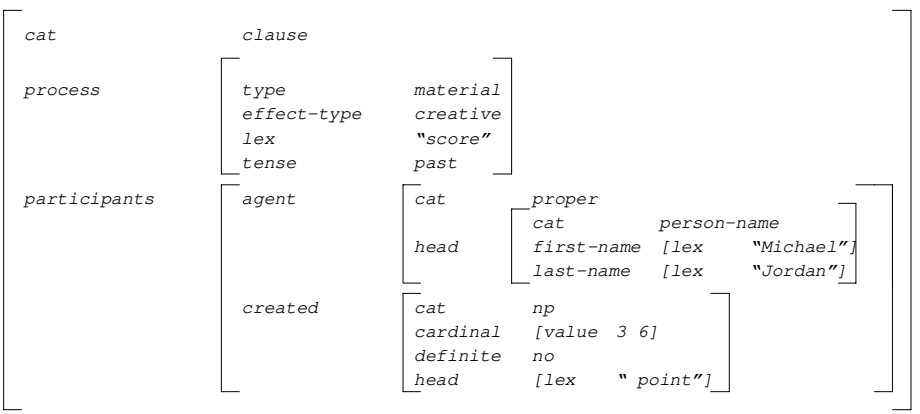

Figure 2. Input for the sentence "Michael Jordan scored 36 points." (Elhadad & Robin, 1996).

dialog systems, they work fine for some systems, but it is difficult to create more sophisticated generation systems using templates. In principle, a template-based system can generate every sentence a rule-based system can generate, if the set of templates covers all of the possible sentences the grammar rules can generate. However, in the general case, that is not very practical.

There are, however, advantages of using a template-based system. One is that the input to a template-based system can be minimally specified in terms of linguistic detail. Instead, the requisite knowledge is embedded implicitly in the templates themselves. Also, because writing templates does not require formal training in computational linguistics, anyone who can create acceptable English (or other target language), typically a literate speaker of the language, can design a template-based system, and if that person can additionally program in a computer language, he/she can implement a complete NLG system without further ado. Reiter (1995) is a more comprehensive comparison of rule-based NLG and template-based NLG.

In contrast, corpus-based techniques provide an alternative to rule- and template-based techniques. Corpus-based techniques attempt to capture the high quality of language used by a domain expert while minimizing the need for explicitly coding linguistic knowledge. The general idea is to use a corpus to automatically learn rules for sentence planning and/or surface realization. Nitrogen (Langkilde & Knight, 1998), a generation engine within a machine translation system, is a rule-based text generation system augmented with a corpus-based technique. Others, such as Oh and Rudnicky (2000), Ratnaparkhi (2000), and Walker, Rambow, and Rogati, 2001 have also shown success with variations of the corpus-based approach.

4. Language generation in the Carnegie Mellon Communicator

The Carnegie Mellon Communicator (Rudnicky *et al.*, 1999) is a telephone-based spoken dialog system. It enables users to arrange complex travel itineraries via natural conversation. It is made up of several modules working together to provide a smooth interaction for the users while completing the task of arranging travel to fit the user's constraints.

SPHINX-II automatic speech recognizer takes the user's speech and outputs a string. The PHOENIX semantic parser takes the string and turns it into a semantic frame. The AGENDA dialog manager (Xu & Rudnicky, 2000) handles the semantic frames from the PHOENIX parser and interacts with additional domain agents such as the date/time module, the user preferences database, and the flight information agent. When the dialog manager decides that something needs to be communicated to the user, it sends a semantic frame to NLG, which then generates an utterance to be synthesized by the TTS module. The input frame from the dialog manager specifies a general "act", which is similar to a speech act, plus a domain-relevant "content" tag. Together these define the utterance class, or the equivalent of a dialog act. The information to be relayed to the listener is specified by attribute-value pairs, some of which may be hierarchically structured. The input frame contains no traditional linguistic features such as subject, patient, tense, head, etc. An example of the input frame is in Figure 3.

The Communicator NLG module is implemented in Perl. These and other implementation details have not changed much since its inception. What has changed, however, is that the system went from a purely template-based generation to corpus-based, stochastic generation.

```
{
  act query
  content depart_time
  depart_date {
                year 2000
                month 10
                day 5
              }
  depart_airport BOS
}
=> What time on October fifth would you like to leave Boston?
```

Figure 3. An input frame to NLG in the Communicator.

4.1. Template-based generation

Our NLG module started off with around 50 templates. The number of templates grew as we added more functionality to our system. The largest expansion came with the addition of a "help" speech act, which added 16 templates to provide context-sensitive help messages. Additional information about the template system is available in Oh (2000). Note that templates are not simple sentence frames with variable slots. They also need to include a computational component that deals with options For example, for the template "What time would you like to travel from {departure_city} on {departure_date}?", if the input frame did not contain values for the attributes {departure_city} and {departure_date}, instead of generating the ungrammatical sentence "What time would you like to travel from on?", it would generate "What time would you like to travel?". This reduces the number of templates significantly, but only at the expense of introducing more complexity to the templates, especially for templates that can have as many as ten different attributes. Hence, the amount of time the developer needs to spend on crafting and maintaining the templates does not decrease significantly. At one point, the number of templates grew to nearly one hundred, some of them very complex and cumbersome to maintain. Axelrod (2000) has alluded to similar requirements in the system that he has described.

4.2. Development of corpus-based stochastic generator

What is perhaps more important than reducing development time is being able to generate utterances that promote a natural interaction with the user. One of the difficulties for a template writer is choosing the right words, the template system's equivalent of lexical selection. Often, the words that the template writer chooses for a given utterance are different from what the domain expert would use. This mismatch may hamper a smooth interaction because when a system utterance contains unfamiliar words in that domain, not only does it sound unnatural, but it may also lead the user to confusion or an inappropriate response.

One solution might be to base the generator on a corpus of task-oriented human–human conversations between a domain expert and a client. We could, for example, take the expert's utterances and use them directly as templates. This is very simple, but is not practical, as one would need to find an utterance for every possible combination of attributes.

The statistical n-gram language model provides an alternative representation. The n-gram language model has the advantage that it is simple to build and understand, and

tools, such as the CMU-Cambridge Language Modeling Toolkit (Clarkson & Rosenfeld, 1997), are readily available. There is precedent in Langkilde and Knight's (1998) application of n-gram models to the problem of smoothing the output of a translation system. A possible objection is that while the n-gram language model captures well the relationships among words in the window of a length specified by the parameter n, it does not capture any long-distance dependencies beyond that window. We argue that this is not an issue for spoken dialog systems because the utterances are simpler in structure, and we can choose the parameter n to optimize performance. We will revisit this issue in a later section.

5. Modeling human–human interaction

Since it is not clear what the "best practices" in designing human–computer spoken interactions are, especially in deciding what the system prompts ought to be, the obvious choice would be to use models of human–human interactions. Boyce and Gorin (1996) support this argument by their definition of a natural dialog: "[a dialog] that closely resembles a conversation two humans might have". Applying this definition to the language generation module of the spoken dialog system, we can build computational models of a domain expert having a conversation with a client, and use those models to generate system utterances.

For many domains, acquiring the correct lexicon items or grammar rules is not a trivial task, and to date, most researchers relied on informal methods of knowledge acquisition. Reiter, Robertson, and Osman (2000), with their recent experiment of structured knowledge acquisition techniques, have begun exploring more principled ways of knowledge acquisition. Although the technique presented here is much simpler than theirs, concentrating mostly on acquisition of lexicon, it can be thought of as an efficient and effective way of automatically acquiring knowledge needed for a flexible language generation system.

5.1. Corpora

When building a statistical model, it is important to choose a data set that will provide statistics most similar to the underlying distribution. In many applications, such as a large vocabulary speech recognition system, this is not trivial. Collecting appropriate data for our generation engine, however, was not problematic. First of all, unlike the speech recognition system where the language model should be able to predict a wide variety of sentences that have never occurred, a language model for our generation engine does not require that predictive power. Second, collecting in-domain data is in any case the first step in building a task-oriented dialog system. Therefore, we can leverage the data collection effort and significantly reduce the amount of labor needed to build a corpus-based generation system.

We used two corpora in the travel reservations domain to build n-gram language models. One corpus (henceforth, the CMU corpus) consists of 39 dialogs between a travel agent and clients (Eskenazi *et al.*, 1999). Another corpus (henceforth, the SRI corpus) consists of 68 dialogs between a travel agent and users in the SRI community (Kowtko & Price, 1989). Table I presents some statistics for the two corpora.

TABLE I. Corpora used and their statistics

	CMU (Agent)	CMU (User)	SRI (Agent)	SRI (User)
No. of dialogs	39		68	
No. of utterances	970	946	2245	2060
No. of words	12 852	7848	27 695	17 995

5.2. Speech act and concept tagging

Instead of just one language model for all the system utterances, we use a language model for each utterance class. This requires that the domain expert's utterances in the training corpora be tagged with the utterance classes. Figure 4 shows the list of utterance classes that was used to tag the corpora.

Another set of tags is necessary for word classes (see Fig. 5). This is similar to the use of word classes in class-based language models, where, for example, words indicating numerals (e.g., one, ten, twelve hundred) are treated as a single unique word in the n-gram model, and then there is a separate model for the distribution of those words in the class. Similarly, in our model, we replace the words in a word class with the tag, treating them as a single unique word for purposes of sentence modeling.

A separate set of class models is not needed, however, since the word classes represent the attributes, for which the values are passed in the input frame from the dialog manager. This is similar to having slots in templates, and then filling the slots with the values in the input.

Much of the tagging effort can be automated. A list of words for most of the word classes is already available as a byproduct of overall system development: city names, airports, airlines, etc. However, as can be seen in Figure 5, some of the word classes

```
query arrive_city          inform airport
query arrive_time          inform confirm_utterance
query confirm              inform epilogue
query depart_date          inform flight
query depart_time          inform flight_another
query pay_by_card          inform flight_earlier
query preferred_airport    inform flight_earliest
query return_date          inform flight_later
query return_time          inform flight_latest
hotel car_info             inform flight_returning
hotel hotel_chain          inform not_avail
hotel hotel_info           inform num_flights
hotel need_car             inform price
hotel need_hotel           other
hotel where
```

Figure 4. Utterance classes.

airline	*depart_date*
am	*depart_time*
arrive_airport	*depart_tod*
arrive_city	*flight_num*
arrive_date	*hotel*
arrive_time	*hotel_city*
car_company	*hotel_price*
car_price	*name*
connect_airline	*num_flights*
connect_airport	*pm*
connect_city	*price*
depart_airport	
depart_city	

Figure 5. Word classes.

need to be more detailed. For example, the class "city" needs to be differentiated into *departure city* (`depart_city`) and *arrival city* (`arrive_city`). Tagging first with the more general class, and then applying simple rules such as "from {city}" becomes "from {depart_city}" produces the desired result. An average of seven rules, each rule consisting of two or three semantically similar word ("depart" and "leave") per word class were used.

Tagging utterance classes requires a little bit more care. Still, it can be semi-automated using machine learning. We first tag several utterances manually, then build trigrams. Then we use the trigram models to tag the rest of the utterances. Using the models from the CMU corpus to automatically tag the SRI corpus, we obtained an average precision of 91% with a recall of 86%.

5.3. Using models of human–human interaction for human–computer interaction

Several issues arise when using computational models of human–human interaction for spoken dialog systems. First, there are some inherent differences between human–human conversations and human–computer conversations. As Boyce and Gorin (1996) point out, there are "user" and "system" utterances in human–computer interactions that do not occur in normal human–human interactions. These include more frequent confirmations, error messages, and help messages. Similarly, there are some utterances that occur frequently in human–human conversations but not in human–computer conversations, the most obvious being backchanneling (e.g., "uh-huh", "okay").

The second issue is that the quality of the output depends very much on the speaker whose language is modeled. This means the selection of a speaker is crucial for system performance. In the case of a travel reservation system, recruiting an experienced travel agent (as we did) may be sufficient, but in other domains it may not be as simple. Another issue is that while the models of human–human interaction may result in natural dialogs, they may not lead to the most efficient dialogs. That is, it may be possible that one could design a computer agent that can carry out a more efficient task-oriented dialog than a human expert. Such issues are beyond the scope of the current paper.

6. Content planning

Content planning is the process by which the system decides which attributes (represented as word classes, see Fig. 5) should be included in an utterance of a given type. In a task-oriented dialog, the number of possible attributes with specified values generally increases during the course of the dialog, as the user introduces his/her constraints and the system contributes information. Therefore, as the dialog progresses, the system needs to decide which ones to include at each system turn. If the system includes all of them every time (indirect echoing, see Hayes & Reddy, 1983), utterances become overly lengthy (see Dialog 1 in Fig. 6), but if we remove all system confirmations, the user receives no feedback about whether his/her utterances were recognized correctly (see Dialog 2 in Fig. 6). Since automatic speech recognition systems still produce significantly more errors than humans, this is an important issue.

In the spoken dialog literature, e.g., Litman and Pan (2001), deciding what to do is embodied in a *confirmation strategy*. Explicitly asking the user to confirm all attributes is an *explicit confirmation* strategy, implicitly confirming the attributes within a system utterance is an *implicit confirmation* strategy, and not confirming any attributes is a *no confirmation* strategy.

Instead of looking at how to incorporate different strategies, we focused on the *implicit confirmation* strategy as being the most compatible with our overall design goals, and focuses on the problem of identifying which attributes to confirm in a particular turn. We compared two ways to systematically generate system utterances with only selected attributes, such that the user hears repetition of some of the constraints he/she has specified, at appropriate points in the dialog, without sacrificing naturalness and efficiency. The specific problems, then, are deciding what should be repeated, and when. We first describe a simple heuristic of old vs. new information. Then we present a statistical approach, based on bigram models.

We present the details of the two approaches in this section, and Section 8.3.1 presents an experiment comparing the approaches.

6.1. Heuristic confirmation

As a simple solution, we can use the previous dialog history, by flagging attribute-value pairs as *old* (previously spoken by the system) information or *new* (not spoken by the system as yet) information. The generation module would select only new information for inclusion in system utterances. Consequently, information given by the user is repeated only once in the dialog, usually in the utterance immediately following the user utterance in which the new information was provided.

	Dialog 1		Dialog 2
A:	Where would you like to go?	A:	Where would you like to go?
U:	I'd like to go from Pittsburgh to San Francisco.	U:	I'd like to go from Pittsburgh to San Francisco.
A:	When would you like to travel from Pittsburgh to San Francisco?	A:	When would you like to travel?
U:	October first	U:	October first
A:	What time on October first would you like to leave Pittsburgh for San Francisco?	A:	What time would you like to leave?
U:	Around 10 a.m.	U:	Around 10 a.m.

Figure 6. Comparison of two extreme strategies.

Although this approach seems to work fairly well, echoing user's constraints a single time may not be the right thing to do. Examining human–human dialogs, we find that this is not what we observe in natural conversations; humans often repeat mutually known information, and they also often do not repeat some information at all. Also, this model does not appear to capture the close relationship between two consecutive utterances within a dialog. The second approach tries to address these issues.

6.2. Stochastic confirmation

We observe that, in a conversation, the content of a turn depends on the content of the previous turn (of the other conversant). For example, if the travel agent has asked about the destination city, the client is more likely to talk about the destination city and perhaps arrival date than, say, departure time. Also, we note that the length of the utterance (i.e., the number of attributes in an utterance) depends on the utterance class. To capture these observations computationally, we built a two-stage statistical model of human–human dialogs using the CMU corpus. The model first predicts the number of attributes in the system utterance given the utterance class, then predicts the attributes included given the attributes found in the previous user utterance.

6.2.1. Attribute number model

The first model will predict the number of attributes in a system utterance given the utterance class, using the following probability distribution

$$P(n_k) = P(n_k \mid c_k), \tag{1}$$

where n_k is the number of attributes and c_k is the utterance class for system utterance k.

6.2.2. Attribute bigram model

This model will predict which attributes to include in a system utterance. Using a statistical model, what we need to do is find the set of attributes $A^* = \{a_1, a_2, \ldots, a_n\}$ in a system utterance such that

$$A^* = \text{argmax} P(a_1, a_2, \ldots, a_n) \tag{2}$$

We assume that the distributions of the a_is are dependent on the attributes in the previous utterances. As a simple model, we look only at the utterance immediately preceding the current utterance and build a bigram model of the attributes. In other words,

$$A^* = \text{argmax} P(A \mid B), \tag{3}$$

where $B = \{b_1, b_2, \ldots, b_m\}$, the set of m attributes in the preceding user utterance.

6.2.3. Combined model

If we took the above model and tried to apply it directly, we would have run into a serious data sparseness problem, so we make two independence assumptions. The first assumption is that the attributes in the user utterance contribute independently to the

probabilities of the attributes in the system utterance following it. Applying this assumption to the model above, we get the following:

$$A^* = \text{argmax} \sum_{k=1}^{m} P(b_k)P(A \mid b_k). \tag{4}$$

The second independence assumption is that the attributes in the system utterance are independent of each other. This gives the final model that we used for selecting the attributes

$$A^* = \text{argmax} \sum_{k=1}^{m} P(b_k) \prod_{i=1}^{n} P(a_i \mid b_k). \tag{5}$$

Although this independence assumption is an oversimplification, this simple model is a good starting point for our initial implementation of this approach.

7. Surface realization

If a natural human–computer dialog is one that closely resembles a human–human conversation, the best method for generating natural system utterances would be to mimic human utterances. In our case, where the system is acting as a travel agent, the solution would be to use a human travel agent's utterances (see Section 5 for details about the training data). The computational model we chose to use is the simple n-gram model familiar from speech recognition (Jelinek, 1998).

7.1. Implementation

We have implemented a hybrid language generation module incorporating three different techniques: fixed templates (e.g., "Welcome to the Carnegie Mellon Communicator"), variable templates (e.g., "Hello *Alice*"), and corpus-based stochastic generation. Certain prompts, for example a system greeting at the outset of the dialog, can be generated by a "canned" expression. Other simple utterances can be generated efficiently by templates. Then, for the remaining utterances where there is a good match between human–human interaction and human–computer interaction (i.e., for a predefined utterance class in the system, there exist utterances in human–human corpora that directly map to the utterance class), we use statistical language models. In the CMU Communicator, approximately half of all utterances produced by the system will be stochastically generated, across typical calls.

There are four aspects to our stochastic surface realizer: building language models, generating candidate utterances, scoring then selecting among the utterances, and filling in the slots. Figure 7 shows how the four components work together, and we describe each in detail below.

7.1.1. Building language models

Using the tagged utterances as described in the introduction, we built an unsmoothed n-gram language model for each utterance class. We settled on 5-gram models to balance variability in the output utterances with the need to minimize the generation of nonsense utterances. In fact utterances acceptable to humans can be generated with 2- and 3-gram models (see the results of a user experiment varying the parameter n in

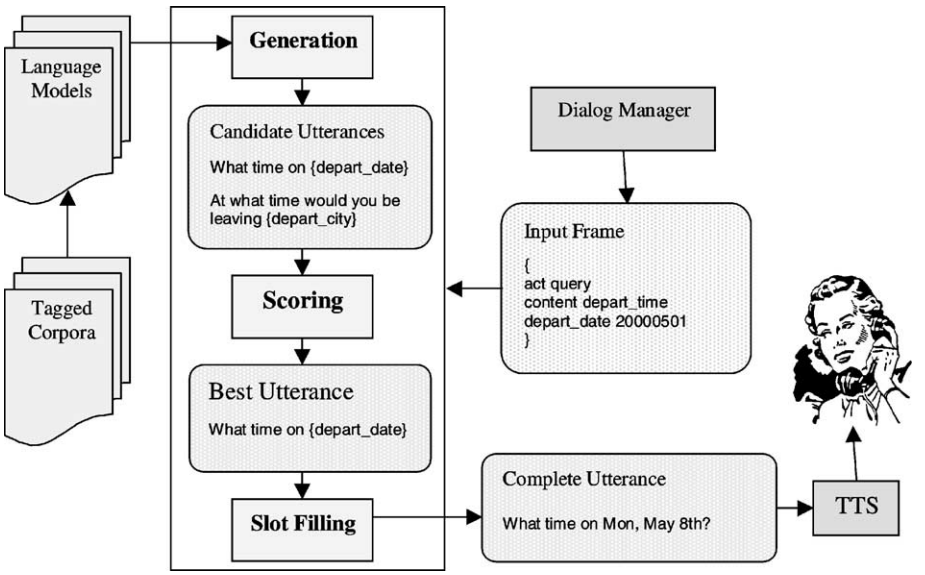

Figure 7. Components of stochastic NLG.

Table II). It is an open question whether smoothing will have a desirable or undesirable effect on the generative use of language models; however it does not appear to us to be a well-formed question: there is indeed a critical need, in recognition, to allow for unseen word sequences (at whatever low probability). There does not appear to be a corresponding need for this property in generation: the models are quite fertile as is (see Table III) and there is no need for novelty per se.

Note that language models are not used here in the same way as in speech recognition. In speech recognition, the language model probability acts as a prior in determining the most probable sequence of words given the acoustics. In other words,

$$W^* = \text{argmax} P(W|A), \tag{6}$$

$$= \text{argmax} P(A|W)P(W), \tag{7}$$

where W is the string of words, $w_1, \ldots, w_n$, and A is the acoustic evidence (Jelinek, 1998).

TABLE II. How 63 subjects rated sentences generated by n-grams (1 is very easy to understand, 5 is very difficult to understand) for the *inform* speech acts and the *query* speech acts

n	Mean (*inform*)	Mean (*query*)
1	4.48	4.53
2	2.25	1.75
3	2.10	1.62
4	1.88	1.48
5	1.86	1.32

TABLE III. The number of unique utterances in the output generated by 5-grams compared with the number of unique utterances in the original corpus

Utterance class	Unique utts in corpus	Unique utts in output
Inform_flight	37	445
Inform_flight_earlier	6	54
Inform_flight_later	9	340
Query_depart_date	22	61
Query_depart_time	20	74

Although we use the same statistical tool to compute the model, we use language model probabilities directly to generate the next word. In other words, the most likely utterance is $W^* = \text{argmax}P(W|u)$, where u is the utterance class. We do not, however, look for the most likely hypothesis, but rather generate each word randomly according to the distribution, as illustrated in the next section.

7.1.2. Generating utterances

The input to NLG from the dialog manager is a frame of attribute-value pairs. The first two attribute-value pairs specify the utterance class. The rest of the frame contains word classes and their values. See Figure 3 for an example of an input frame to NLG.

The generation engine uses the appropriate language model for the utterance class and generates word sequences randomly according to the language model distributions. As in speech recognition, the probability of a word using the n-gram language model is

$$P(w_i) = P(w_i | w_{i-1}, w_{i-2}, \ldots, w_{i-(n-1)}, u), \tag{8}$$

where u is the utterance class. Since we have built separate models for each of the utterance classes, we can ignore u, and say that

$$P(w_i) = P(w_i | w_{i-1}, w_{i-2}, \ldots, w_{i-(n-1)}) \tag{9}$$

using the language model for utterance class u.

Stochatic generation is highly fertile, as seen in the number of distinct utterances generated for two representative speech acts (one simple, one complex), shown in Table III. There is the problem, as in speech recognition using n-gram language models, that long-distance dependency cannot be captured. However, this does not appear to significantly affect the comprehensibility of the utterances.

7.1.3. Scoring utterances

For each randomly generated utterance, we compute a penalty score. The score is based on the following heuristic rules. Penalties are assigned for the following:

1. The utterance is too short or too long (determined by an expectation derived from the length observed for that class of utterance in the corpus).
2. The utterance contains repetitions of any of the slots.
3. The utterance contains slots for which there is no valid value in the frame.
4. The utterance lacks some of the required slots (see Section 6 for deciding which slots are required).

The generation engine generates a candidate utterance, scores it, keeping only the best-scored utterance up to that point. It stops and returns the best utterance when it finds an utterance with a zero penalty score, or it reaches the limit of 50 iterations. The average generation for the longest utterance class (10–20 words long) is about 200 ms (on a 400 MHz Pentium computer).

This search strategy is depth-first, where the full sentence is $w = \{w_1, \ldots, w_n\}$, the search tree depth is n, and each word w_i is a path down the search tree. A similar corpus-based system in Ratnaparkhi (2000) employs breadth-first search.

7.1.4. Filling slots

The last step is filling slots with the appropriate values from the input frame. For example, the utterance "What time would you like to leave {depart_city}?" becomes "What time would you like to leave New York?". A final filter smooths local agreement (e.g., *a American flight* → *an American flight*).

8. Evaluation and discussion

It is generally difficult to evaluate an NLG system, and although more attention has been given to evaluation in the recent years, several issues remain (see Mellish & Dale, 1998). In the context of spoken dialog systems, evaluation becomes even more difficult. One reason is simply that there has been little effort in building sophisticated generation engines for spoken dialog systems, and much less in evaluating such engines. Another reason is that it is difficult to separate the NLG module from the rest of the dialog system, especially its often closely coupled neighbor, text-to-speech synthesis (TTS).

This section discusses briefly some previous evaluation methods in language generation and spoken dialog systems, presents a series of experiments we have conducted for our stochastic generator, and summarizes some issues in evaluating the generation component of a spoken dialog system.

8.1. Evaluation methods for text generation

Several techniques have been proposed to evaluate text generation systems, but there is still a lack of consensus on what constitutes a good evaluation method. Whereas natural language understanding researchers can use resources such as the Penn Treebank (Marcus, Santorini, & Marcinkiewicz, 1993) to compare a parser's output to a generally accepted reference parse, a similar approach is not appropriate for automatic evaluation of generation because there is no one "correct sentence" that generation can produce: many acceptable alternatives are possible. Hence, many evaluation methods rely on human judgment of accuracy, fluency, and readability of the generated text. The methodologies used include user surveys, measuring comprehension/response times, or comparison of human-generated output and machine-generated output. A good survey of evaluation methods is presented in Mellish and Dale (1998).

For corpus-based generation systems, Bangalore, Rambow, and Whittaker (2000) have proposed string-based and tree-based automatic metrics that compare the output sentences with sentences in the original corpus. They show that the tree-based metrics correlate with human judgments and thus are useful for evaluation during development.

However, they add that these automatic metrics cannot replace human judgment, and that the metrics may penalize correct variations of the original sentences.

8.2. Evaluation methods for spoken dialog systems

NLG in spoken dialog systems can be evaluated on several dimensions. The first is an analysis of the technique, which may include grammar coverage, completeness of lexicon, variety of generated utterances, as well as other analytical measures. This dimension plays a major part in building the NLG module, and is an important evaluation criterion. However, we consider this more as a part of system development and maintenance, so we will focus this section on the other dimensions.

The second measure is the quality of the generated output within the embedded system, in terms of accuracy, naturalness, and efficiency. This dimension is probably the major concern for any evaluation metric, and will be the major topic of this section.

The last dimension is the effect of the output utterances on user's responses. This is important for the performance of the dialog system as a whole. Not only does the system output need to be comprehensible, it must elicit a proper response from the user. Of course, this dimension is not independent of the previous dimension, as being able to understand the system utterances is a necessary precondition for the users to correctly respond.

PARADISE (Walker, Litman, & Kamm, 1998) is an effective evaluation framework for spoken dialog systems that attempts to correlate certain properties of a dialog (e.g., ASR performance) with some objective function (e.g., task completion). The measures that contribute most to the objective function would then be targeted to improve the system performance. However, this framework is difficult to apply to an individual component of a spoken dialog system, such as NLG.

Rambow, Rogati, and Walker (2001) present an evaluation scheme for NLG in spoken dialog systems. They asked human subjects to read dialogs generated by different NLG systems and rate the sentences on a 5-point scale for whether the sentences were "easy to understand", "well-formed", and "appropriate to the dialog context". We explored this paradigm in one of our experiments (see Section 8.3.2), but concluded that this paradigm is inadequate, for two reasons.

1. Reading the system utterances is different from listening to them. One might rate a written sentence as "easy to understand", but then one might listen to the same sentence and rate it as "hard to understand". Similarly a listener might find a sentence clear and understandable, while a reader might focus on relatively minor grammatical faults.
2. With this paradigm, it is impossible to evaluate the effect of NLG on the whole dialog system. We feel that, since NLG is part of the dialog system, one of its main objectives should be to increase system performance. Just because the system output is "well-formed" or "easy to understand", it may not translate directly to high system performance.

For these reasons, we felt it would be best to run comparative evaluations, in which subjects run through two sets of dialogs, with only the NLG component changed.

8.3. Comparative evaluation

We conducted this comparative evaluation by running two identical systems varying only the generation component. In this section, we present results from two preliminary evaluations of our generation algorithms described in the previous sections.

8.3.1. Content planning: experiment

For the content planning part of the generation system, we conducted a comparative evaluation of the two different generation algorithms: old/new and bigrams. Twelve subjects had two dialogs each, one with the old/new generation system, and another with the bigrams generation system (in counterbalanced order); all other modules were held fixed. Afterwards, each subject answered eleven questions on a usability survey (see Fig. 8). The first seven questions asked the subjects to compare the two systems, and the remaining four questions asked for user's information and other comments. Immediately after filling out the survey, each subject was given transcribed logs of his/her dialogs and asked to rate each system utterance on a scale of 1 to 3 (1 = good; 2 = okay; 3 = bad).

Table IV shows the results from the usability survey. The results seem to indicate subjects' preference for the old/new system, but as one can tell from looking at the large variances, the difference is not statistically significant. From this we can conclude that heuristics and bigrams work equally well. However, it is worth noting that six out of the 12 subjects chose the bigram system to the question "During the session, which system's responses were easier to understand?" compared to three subjects choosing the old/new system. We can make an interesting conjecture that since we used the statistical models of

1. *Which system did you prefer to use?*
2. *Which system do you feel gave you what you wanted?*
3. *During the session, which system seemed to understand you better?*
4. *During the session, which system made it clearer to you what you could say next?*
5. *During the session, which system's responses were easier to understand?*
6. *During the session, which system offered you more information?*
7. *If you were to use this type of system again, which of the two would you choose?*
8. *How many times have you used this system before this test?*
9. *What did you like the most about these systems?*
10. *What did you like the least about these systems?*
11. *Please write any other comments you have.*

Figure 8. Questions on user survey.

human–human conversations to generate the content of the system utterances, users found those utterances easier to understand.

8.3.2. Surface realization: experiment 1

For surface realization, we first conducted a batch-mode evaluation. We picked six calls to the CMU Communicator and ran two generation algorithms (template-based generation and stochastic generation) on the input frames. We then presented to seven subjects the transcripts of the generated dialogs, consisting of decoder output of the user utterances and corresponding system responses, for each of the two generation algorithms. Subjects then selected the output utterance they would prefer, for each of the utterances that differ between the two systems. The results in Table V show a trend that subjects preferred stochastic generation to template-based generation, although a t test for all the subjects shows no significant difference ($p = 0.18$).

These results, while encouraging, are inconclusive. While that may be an acceptable proof that our simple stochastic generator performs at least as well as a carefully handcrafted template system, we wished to conduct another experiment with more subjects. Another problem with this experimental setup was that subjects were reading written transcripts of the dialogs, rather than listening to the dialogs and evaluating them. We set it up that way to simplify the experiment, but our concern after looking at the results was that subjects' judgments were tainted by the different modality. Hence, we tried another experiment where subjects both read and listened to the dialogs.

8.3.3. Surface realization: experiment 2

We conducted this experiment via the web to make it easier for more subjects to participate. As in the first experiment, we picked three dialogs and ran them through the two different generation engines. In this second experiment, we also wished to use audio files of the dialogs. To evaluate our experimental methodology, we also asked the subjects to judge the transcripts of the same dialogs. The set-up was as follows:

1. The subject receives an email message containing a link to the webpage.
2. The first webpage describes the experiment and gives instructions. It also contains a link to the experiment.
3. The experiment page has links to the audio (.wav) files where the subjects can click and listen to the dialogs.

TABLE IV. Means and variances of the 12 users' answers on the usability survey on the first seven questions (−1 is preferring the old/new system, +1 is preferring the bigram system, and 0 is neither or both)

Question	Mean	Var
Preference	−0.08	0.63
Task completion	−0.08	0.45
Understand user	−0.08	0.81
Know what to say next	−0.17	0.52
Easier to understand	0.25	0.75
More information	−0.08	0.63
Future use	0.00	0.55

TABLE V. User preferences (49 utterances total)

Subject	Stochastic	Templates	Difference
1	41	8	33
2	34	15	19
3	17	32	−15
4	32	17	15
5	30	17	13
6	27	19	8
7	8	41	−33
Average	27	21.29	5.71

4. After listening to two different versions of each of the dialogs (3 pairs of dialogs were used), the subject answers a set of questions (see Fig. 9).
5. After rating all three pairs of dialogs, the subject clicks to go onto the next webpage.
6. The webpage contains transcripts of the dialogs the subject just heard. This time, the subject reads the transcripts and picks, for each dialog pair, which one is better.

For both the audio and transcript evaluations, the results again showed us that both stochastic and template systems performed equally well. Table VI shows the results of the questions after listening to the audio files. If we look at the scores for subjects individually, 11 subjects gave better scores for stochastic, 8 for templates, and 1 neutral, but of those, only 6 were statistically significant (3 for stochastic, 3 for templates). Looking at the results from the transcript evaluation, 11 subjects preferred stochastic, 7 preferred templates, and 3 were neutral. Of those, only 2 (both for stochastic) were statistically significant. As for the correlation between the two modalities, there seemed to be no correlation ($r = 0.009$). This tells us that, in fact, reading the dialogs leaves very different impressions from listening to the dialogs, and since we are evaluating an NLG component within a spoken dialog system, having the subjects read and rate

1. *Which dialog was more natural? Why?*
2. *Which dialog did you understand better? Why?*
3. *Which dialog did you like better? Why?*
4. *Which system would you prefer to use? Why?*

Figure 9. Questions after each pair of dialogs.

TABLE VI. Subjects' preferences on questions after listening to the dialogs (+1 for stochastic, −1 for templates)

Question	Mean	*p*-Value
Natural	−0.02	0.43
Understand	0.08	0.14
Like better	0.12	0.13
Prefer to use	−0.02	0.43

sentences may not be accurate. All in all, the results from all of the experiments show that the stochastic generator does as well as the template system.

9. Conclusion

We felt that the popular techniques in NLG are not adequate for spoken dialog systems, and explored an alternative approach. The aim was to create a high quality generation system without having to have an expert grammar writer or the cumbersome maintenance of templates. We leveraged the data collection effort done for other parts of the dialog system, tagged the data, and built stochastic models of human–human interaction. By this corpus-based approach, we were able to capture the domain expert's knowledge into simple n-gram models.

By augmenting our template system with this stochastic method, we built a hybrid system that produces high quality output without the need for complex grammar rules and with minimally specified input. For both content planning and surface realization, we showed that the stochastic method perform as well as the carefully crafted template system. There also seemed to be a trend for subjects to prefer the stochastic system, but this preference was not statistically significant.

We conducted experiments to evaluate our system, and by doing so, we discovered various elements one must be careful about when designing these experiments. Teasing apart NLG from the dialog system is difficult, but with our comparative evaluation scheme, it is possible to do so. We believe this will open the door to further progress on evaluation of NLG in spoken dialog systems.

We would like to thank the members of the Carnegie Mellon Speech Group and in particular the contributors to the Communicator project, without whom this work would not have been possible to carry out. We would also like to acknowledge the contribution of Lea of People's Travel in Pittsburgh, PA, US, who helped us create the necessary corpus of travel agent / client interactions. Portions of this work were first described at the April 2000 NAACL Workshop on Dialog Processing and, earlier, in presentations at principal investigator meetings of the DARPA Communicator Program. This research was sponsored in part by the Space and Naval Warfare Systems Center, San Diego, under Grant No. N66001-99-1-8905. The content of the information in this publication does not necessarily reflect the position or the policy of the US Government, and no official endorsement should be inferred.

References

Axelrod, S. (2000). Natural language generation in the ibm flight information system. *Proceedings of ANLP/NAACL 2000 Workshop on Conversational Systems*, pp. 21–26.

Bangalore, S., Rambow, O. & Whittaker, S. (2000). *Evaluation metrics for generation. Proceedings of the International Conference on Natural Language Generation.*

Baptist, L. & Seneff, S. (2000). *Genesis-ii: a versatile system for language generation in conversational system applications. Proceedings of International Conference on Spoken Language Processing.*

Bateman, J. & Henschel, R. (1999). *From full generation to 'near-templates' without losing generality. Proceedings of the KI'99 workshop* "May I Speak Freely?".

Boyce, S. & Gorin, A. (1996). User interface issues for natural spoken dialog systems. *Proceedings of the International Symposium on Spoken Dialog*, pp. 65–68.

Busemann, S. & Horacek, H. (1998). A flexible shallow approach to text generation. In *Proceedings of the Ninth International Workshop on Natural Language Generation* (E. Hovy, Ed), pp. 238–247. Association for Computational Linguistics, New Brunswick, NJ.

Clarkson, P. & Rosenfeld, R. (1997). *Statistical language modeling using the cmu-cambridge toolkit. Proceedings of Eurospeech97.*

Elhadad, M., Robin, J. (1996). An overview of surge: A reusable comprehensive syntactic realization component. Technical Report 96-03, Department of Mathematics and Computer Science.

Eskenazi, M., Rudnicky, A., Gregory, K., Constantinides, P., Brennan, R., Bennett, C. & Allen, J. (1999). Data collection and processing in the Carnegie Mellon Communicator. *Proceedings of Eurospeech* **6**, 2695–2698.

Gorin, A. L., Riccardi, G. & Wright, J. H. (1997). How may I help you? *Speech Communication* **23**, 113–127.

Hayes, P. J. & Reddy, D. R. (1983). Steps toward graceful interaction in spoken and written man–machine communication. *International Journal of Man–Machine Studies* **19**, 231–284.

Jelinek, F. (1998). *Statistical Methods for Speech Recognition*, The MIT Press, Cambridge, MA.

Kotelli, B. (2001). *Blancusi, neo-plasticism, and the art of speech-recognition applications. Proceedings of the Automatic Speech Recognition and Understanding Workshop.*

Kowtko, J. & Price, P. (1989). *Data collection and analysis in the air travel planning domain. Proceedings of DARPA Speech and Natural Language Workshop.*

Langkilde, I. & Knight, K. (1998). *The practical value of n-grams in generation. Proceedings of International Natural Language Generation Workshop.*

Litman, D. & Pan, S. (2001). Designing and evaluating an adaptive spoken dialogue system. *User Modeling and User-Adapted Interaction.*

Marcus, M., Santorini, B. & Marcinkiewicz, M. A. (1993). Buidling a large annotated corpus of English: the Penn treebank. *Computational Lingusitics*, 19.

Mellish, C. & Dale, R. (1998). Evaluation in the context of natural language generation. *Computer Speech and Language* **12**, 349–373.

Oh, A. (2000). Stochastic language generation for spoken dialog systems. Carnegie Mellon University, Masters Thesis.

Oh, A. & Rudnicky, A. (2000). Stochastic language generation for spoken dialog systems. *ANLP/NAACL Workshop on Conversational Systems*, 27–32.

Rambow, O., Rogati, M. & Walker, M. (2001). Evaluating a trainable sentence planner for a spoken dialogue travel system. *Meeting of the Association of Computational Linguistics.*

Ratnaparkhi, A. (2000). Trainable methods for surface natural language generation. *Proceedings of the ANLP/NAACL 2000 Workshop on Conversational Systems*, pp. 194–201.

Reiter, E. (1995). Nlg vs. templates. *Proceedings of the Fifth European Workshop in Natural Language Generation*, pp. 95–105.

Reiter, E., Robertson, R. & Osman, L. (2000). Knowledge acquisition for natural language generation. *Proceedings of the First International Natural Language Generation Conference*, pp. 217–224.

Rudnicky, A., Thayer, E., Constantinides, P., Tchou, C., Shern, R., Lenzo, K., Xu, W. & Oh, A. (1999). Creating natural dialogs in the Carnegie Mellon Communicator system. *Proceedings of Eurospeech* **4**, 1531–1534.

Stent, A. (1999). *Content planning and generation in continuous-speech spoken dialog systems. Proceedings of the KI'99 workshop* "May I Speak Freely?".

Walker, M., Litman, D. & Kamm, C. (1998). Evaluating spoken dialogue agents with PARADISE: two case studies. *Computer Speech and Language*,, 3–12.

Walker, M., Rambow, O. & Rogati, M. (2001). *Spot: a trainable sentence planner. Proceedings of the North American Meeting of the Association of Computational Linguistics.*

Xu, W. & Rudnicky, A. (2000). Task-based dialog management using an agenda. *Proceedings of ANLP/NAACL 2000 Workshop on Conversational Systems*, pp. 42–47.

Zue, V. (2000). Jupiter: a telephone-based conversational interface for weather information. *IEEE Transactions on Speech and Audio Processing* **8(1)**.

(Received 5 October 2001 and accepted for publication 26 April 2002)