Stochastic Language Generation for Spoken Dialogue Systems

Alice H. Oh
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
aliceo+@cs.cmu.edu

Alexander I. Rudnicky
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
air+@cs.cmu.edu

Abstract

The two current approaches to language generation, template-based and rule-based (linguistic) NLG, have limitations when applied to spoken dialogue systems, in part because they were developed for text generation. In this paper, we propose a new corpus-based approach to natural language generation, specifically designed for spoken dialogue systems.

Introduction

Several general-purpose rule-based generation systems have been developed, some of which are available publicly (cf. Elhadad, 1992). Unfortunately these systems, because of their generality, can be difficult to adapt to small, task-oriented applications. 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 proposed a similar approach for a spoken dialogue system. However, there is still the burden of writing and maintaining grammar rules, and processing time is probably too slow for sentences using grammar rules (only the average time for templates and rule-based sentences combined is reported in Busemann and Horacek, 1998), for use in spoken dialogue systems.

Because comparatively less effort is needed, many current dialogue systems use template-based generation. But there is one obvious disadvantage: the quality of the output depends entirely on the set of templates. Even in a relatively simple domain, such as travel reservations, the number of templates necessary for reasonable quality can become quite large that maintenance becomes a serious problem. There is an unavoidable trade-off between the amount of time and effort in creating and maintaining templates and the variety and quality of the output utterances.

Given these shortcomings of the above approaches, we developed a corpus-based generation system, in which we model language spoken by domain experts performing the task of interest, and use that model to stochastically generate system utterances. We have applied this technique to sentence realization and content planning, and have incorporated the resulting generation component into a working natural dialogue system (see Figure 1). In this paper, we describe the technique and report the results of two evaluations.

We used two corpora in the travel reservations domain to build n-gram language models. One corpus (henceforth, the CMU corpus) consists of 39 dialogues between a travel agent and clients (Eskenazi, et al. 1999).

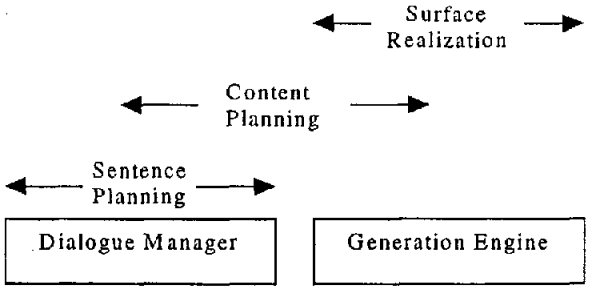

Figure 1 : Overall Architecture

query_arrive_city	inform_airport
query_arrive_time	inform_confirm_utterance
query_confirm	inform_flight
query_depart_date	inform_flight_another
query_depart_time	inform_flight_earlier
query_pay_by_card	nform_flight_earliest
query_preferred_airport	inform_flight_later
query_return_date	inform_flight_latest
query_return_time	inform_not_avail
hotel_car_info	inform_num_flights
hotel_hotel_chain	inform_price
hotel_hotel_info	other

Figure 2 : utterance classes

airline	depart_date
arrive_airport	depart_time
arrive_city	flight_num
arrive_date	hotel_city
arrive_time	hotel_price
car_company	name
car_price	num_flights
depart_airport	pm
depart_city	price

Figure 3 : word classes

Another corpus (henceforth, the SRI corpus) consists of 68 dialogues between a travel agent and users in the SRI community (Kowtko and Price 1989).

The utterances in the two corpora were tagged with utterance classes and word classes (see Figure 2 and Figure 3). The CMU corpus was manually tagged, and back-off trigram models built (using Clarkson and Rosenfeld, 1997). These language models were used to automatically tag the SRI corpus; the tags were manually checked.

1 Content Planning

In content planning we decide which attributes (represented as word classes, see Figure 3) should be included in an utterance. In a task-oriented dialogue, the number of attributes generally increases during the course of the dialogue. Therefore, as the dialogue progresses, we need to decide which ones to include at each system turn. If we include all of them every time (indirect echoing, see Hayes and Reddy, 1983), the utterances become overly lengthy, but if we remove all unnecessary attributes, the user may get confused. With a fairly high recognition error rate, this becomes an even more important issue.

The problem, then, is to find a compromise between the two. 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 dialogue, 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 versus new information. Then we present a statistical approach, based on bigram models.

1.1 First approach: old versus new

As a simple solution, we can use the previous dialogue history, by tagging the attribute-value pairs as old (previously said by the system) information or new (not said by the system yet) information. The generation module would select only new information to be included in the system utterances. Consequently, information given by the user is repeated only once in the dialogue, usually in the utterance immediately following the user utterance in which the new information was given[1].

Although this approach seems to work fairly well, echoing user's constraints only once may not be the right thing to do. Looking at human-human dialogues, we observe that this is not very natural for a conversation; humans often repeat mutually known information, and they also often do not repeat some information at all. Also, this model does not capture the close relationship between two consecutive utterances within a dialogue. The second approach tries to address these issues.

1.2 Second approach: statistical model

For this approach, we adopt the first of the two sub-maxims in (Oberlander, 1998) "Do the human thing". Oberlander (1998) talks about generation of referring expressions, but it is universally valid, at least within natural language generation, to say the best we can do is

[1] When the system utterance uses a template that does not contain the slots for the new information given in the previous user utterance, then that new information will be confirmed in the next available system utterance in which the template contains those slots.

to mimic human behavior. Hence, we built a two-stage statistical model of human-human dialogues using the CMU corpus. The model first predicts the number of attributes in the system utterance given the utterance class, then predicts the attributes given the attributes in the previous user utterance.

1.2.1 The number of attributes model

The first model will predict the number of attributes in a system utterance given the utterance class. The model is the probability distribution $P(n_k) = P(n_k|c_k)$, where n_k is the number of attributes and c_k is the utterance class for system utterance k.

1.2.2 The bigram model of the attributes

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

$$A^* = \arg\max \prod P(a_1, a_2, \ldots, a_n)$$

We assume that the distributions of the a_i's 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, $A^* = \arg\max P(A|B)$, where $B = \{b_1, b_2, \ldots, b_m\}$, the set of m attributes in the preceding user utterance.

If we took the above model and tried to apply it directly, we would 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:

$$A^* = \arg\max \sum_{k=1}^{m} P(b_k)P(A \mid b_k)$$

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.

$$A^* = \arg\max \sum_{k=1}^{m} P(b_k) \prod_{i=1}^{n} P(a_i \mid b_k)$$

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

2 Stochastic Surface Realization

We follow Busemann and Horacek (1998) in designing our generation engine with "different levels of granularity." The different levels contribute to the specific needs of the various utterance classes. For example, at the beginning of the dialogue, a system greeting can be simply generated by a "canned" expression. Other short, simple utterances can be generated efficiently by templates. In Busemann and Horacek (1998), the remaining output is generated by grammar rules. We replace the generation grammar with a simple statistical language model to generate more complex utterances.

There are four aspects to our stochastic surface realizer: building language models, generating candidate utterances, scoring the utterances, and filling in the slots. We explain each of these below.

2.1 Building Language Models

Using the tagged utterances as described in the introduction, we built an unsmoothed n-gram language model for each utterance class. Tokens that belong in word classes (e.g., "U.S. Airways" in class "airline") were replaced by the word classes before building the language models. We selected 5 as the n in n-gram to introduce some variability in the output utterances while preventing nonsense utterances.

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,

$$\begin{aligned} W^* &= \arg\max P(W|A) \\ &= \arg\max P(A|W)Pr(W) \end{aligned}$$

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

Although we use the same statistical tool, we compute and use the language model probability directly to predict the next word. In other words, the most likely utterance is $W^* =$

arg max 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.

2.2 Generating Utterances

The input to NLG from the dialogue 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. Figure 4 is an example of an input frame to NLG.

```
{
act-query
content depart_time
depart_city New York
arrive_city San Francisco
depart_date 19991117
}
```

Figure 4 : 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

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

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

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

using the language model for **u**.

Since we use unsmoothed 5-grams, we will not generate any unseen 5-grams (or smaller n-grams at the beginning and end of an utterance). This precludes generation of nonsense utterances, at least within the 5-word window. Using a smoothed n-gram would result in more randomness, but using the conventional back-off methods (Jelinek 1998), the probability mass assigned to unseen 5-grams would be very small, and those rare occurrences of unseen n-grams may not make sense anyway. There is the problem, as in speech recognition using n-gram language models, that long-distance dependency cannot be captured.

2.3 Scoring Utterances

For each randomly generated utterance, we compute a penalty score. The score is based on the heuristics we've empirically selected. Various penalty scores are assigned for an utterance that 1. is too short or too long (determined by utterance-class dependent thresholds), 2. contains repetitions of any of the slots, 3. contains slots for which there is no valid value in the frame, or 4. does not have some required slots (see section 2 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 runs out of time.

2.4 Filling Slots

The last step is filling slots with the appropriate values. For example, the utterance "What time would you like to leave {depart_city}?" becomes "What time would you like to leave New York?".

3 Evaluation

It is generally difficult to empirically evaluate a generation system. In the context of spoken dialogue systems, evaluation of NLG becomes an even more difficult problem. One reason is simply that there has been very little effort in building generation engines for spoken dialogue systems. Another reason is that it is hard to separate NLG from the rest of the system. It is especially hard to separate evaluation of language generation and speech synthesis.

As a simple solution, we have conducted a 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.

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 dialogues 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 seven questions on a usability survey. Immediately after, each subject was given transcribed logs of his/her dialogues and asked to rate each system utterance on a scale of 1 to 3 (1 = good; 2 = okay; 3 = bad).

3.2 Content Planning: Results

For the usability survey, the results seem to indicate subjects' preference for the old/new system, but the difference is not statistically significant ($p = 0.06$). However, six out of the twelve 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.

3.3 Surface Realization: Experiment

For surface realization, we conducted a batch-mode evaluation. We picked six recent calls to our system and ran two generation algorithms (template-based generation and stochastic generation) on the input frames. We then presented to seven subjects the generated dialogues, 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 show a trend that subjects preferred stochastic generation over template-based generation, but a t-test shows no significant difference ($p = 0.18$). We are in the process of designing a larger evaluation.

4 Conclusion

We have presented a new approach to language generation for spoken dialogue systems. For content planning, we built a simple bigram model of attributes, and found that, in our first implementation, it performs as well as a heuristic of old vs. new information. For surface realization, we used an n-gram language model to stochastically generate each utterance and found that the stochastic system performs at least as well as the template-based system.

Our stochastic generation system has several advantages. One of those, an important issue for spoken dialogue systems, is the response time. With stochastic surface realization, the average generation time for the longest utterance class (10 – 20 words long) is about 200 milliseconds, which is much faster than any rule-based systems. Another advantage is that by using a corpus-based approach, we are directly mimicking the language of a real domain expert, rather than attempting to model it by rule. Corpus collection is usually the first step in building a dialogue system, so we are leveraging the effort rather than creating more work. This also means adapting this approach to new domains and even new languages will be relatively simple.

The approach we present does require some amount of knowledge engineering, though this appears to overlap with work needed for other parts of the dialogue system. First, defining the class of utterance and the attribute-value pairs requires care. Second, tagging the human-human corpus with the right classes and attributes requires effort. However, we believe the tagging effort is much less difficult than knowledge acquisition for most rule-based systems or even template-based systems. Finally, what may sound right for a human speaker may sound awkward for a computer, but we believe that mimicking a human, especially a domain expert, is the best we can do, at least for now.

Acknowledgements

We are thankful for significant contribution by other members of the CMU Communicator Project, especially Eric Thayer, Wei Xu, and Rande Shern. We would like to thank the subjects who participated in our evaluations. We also extend our thanks to two anonymous reviewers.

References

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

Busemann, S. and Horacek, H. (1998) A flexible shallow approach to text generation. In *Proceedings of the International Natural* Language *Generation Workshop.* Niagara-on-the-Lake, Canada.