WHERE DO THE WORDS COME FROM? LEARNING MODELS FOR WORD CHOICE AND ORDERING FROM SPOKEN DIALOG CORPORA

Amanda J. Stent, Srinivas Bangalore, Giuseppe Di Fabbrizio

AT&T Labs – Research
180 Park Ave,
Florham Park, NJ 07932, USA
{stent,srini,pino}@research.att.com

ABSTRACT

Most existing generation systems for spoken dialog require the system engineer to specify by hand the words to be used in system prompts. However, the existence of corpora of spoken dialog makes it possible to acquire the words and structure of system prompts automatically. In this paper, we construct statistical models for generating system prompts, both for word choice and for word ordering. We evaluate these models using a human-computer dialog multicorpus and a human-human dialog corpus. Our results show that statistical models for word choice can work well, while more work is needed on statistical models for word ordering.

Index Terms— language generation, spoken dialog systems, natural language interfaces, machine learning

1. INTRODUCTION

Spoken dialog systems use one of two approaches to generation of system prompts: template-based generation (e.g. [1, 2, 3, 4]), or spoken language generation [5, 6, 7]. A template-based generator typically contains a set of triples of the form {dialog act, content constraints, response template}. Each response template is a string, potentially containing gaps where named entities, numbers, and dates can be filled in. At run time, the generator selects a template to instantiate based on the input dialog act and content. By contrast, a spoken language generator is a pipeline of components that generates system prompts and presentations from first principles. A typical spoken language generator contains a content planner (selecting content from a database), a text planner (that arranges the content using a model of discourse), a sentence planner (which assigns content to sentential units, inserts discourse cues and may perform referring expression generation), and a surface realizer (which structures each sentence, ordering the words it contains and inserting function words if necessary, and performs morphological assignment) [6].

In both generation approaches, the task of selecting words falls to the system engineer. In a template-based generator, word choice is part of template construction. In a spoken language generator, word choice is split between the sentence planner and surface realizer; typically, the sentence planner uses templates or rules to select content words and perform referring expression generation, while the surface realizer may insert function words [7]. Because both approaches rely on human effort for word choice, the generation system typically has limited ability to produce variations on system prompts. Furthermore, the generation context (including previously-made decisions about tense, verb choice, directionality of movement verbs, etc.) is not taken into account during word choice or surface realization. Finally, there is little opportunity for the choice of syntactic structure to influence the choice of words and vice versa (for example, choosing a verb indicating manner of movement vs. indicating it with an adverbial, as in *I will fly there* vs. *I will go by plane*).

In this paper, we describe statistical models for performing word choice and word ordering for response generation in spoken dialog systems. We show the results of evaluations of these models on a multicorpus of human-computer dialogs in the air travel domain and a corpus of human-human dialogs in a catalog ordering domain. We conclude with a discussion of our ideas for improving these models.

2. RELATED WORK

There are essentially three approaches to trainable surface realization: two-stage surface realizers [8, 9, 10], classification-based surface realizers [11, 12], and surface realizers that use probabilistic grammars [13, 14, 15, 16]. Each type of surface realizer uses a statistical model or set of models that capture word order and syntactic constraints (e.g. agreement). All of these surface realizers require at least the content words for the output sentence to be included in the input. In existing spoken language generators, this typically means that the sentence or content planner has to select the content words using a set of sentence or phrase templates (e.g. [7]).

[17] use reinforcement learning to select one of four hand-designed prompts at a decision point in a call routing application. In effect, their model selects the actual utterance for the system prompt.

In recent work [18] learn a generation lexicon and some syntactic structures by mining semantically relevant sentences from a large corpus of user reviews of restaurants and hotels. The resulting lexicon and grammar are used to produce utterances adapted to the reader's or hearer's preferences. Our models are trained on spoken dialog rather than on text data, and so we can use features from the dialog history to condition our choice of words and templates. We also explore a larger range of models than [18].

3. MODELS

If $W = w_1, w_2 \ldots, w_n$ is the sequence of words to be generated and H is contextual information (such as that shown in Table 1), then we model the generation problem (shown in equation 1) as a search problem for the best string of words (W^*) which maximizes the probability $P(W|H)$.

Feature type	Communicator	CHILD
Lexical	word n-grams of contextual features	word n-grams of contextual features
Semantic	Named entities	-
Discourse	Dialog act, task/subtask, conversational domain	Dialog act, task/subtask
Contextual	1 to 3 previous turns by the user 1 previous turn by the system previous utterances in current system turn	1 to 3 previous turns by the customer 1 previous turn by the agent previous utterances in current system turn

Table 1. Features used to train classifiers

$$W^* = \underset{W}{argmax}\ P(W|H) \quad (1)$$

Because there are lots of possible strings of words, we cluster the word strings (C). We investigate a few methods for forming clusters that combine different degrees of word choice and ordering. In the first method, **utterance-string [US]**, we group utterances that have the same content words and the same order of words into a cluster. In the second method, **utterance-bag [UB]**, we disregard the order of words and group utterances that have the same content words into a cluster. These two methods of clustering do not generalize over domain entities (such as names of cities in the air travel domain). We replace named entities in utterances to get templates, and then cluster the templates with (**template-string [TS]**) or without (**template-bag [TB]**) the order information among tokens. Finally, (**dialogact*subtask [DA*ST]**), we use the dialog act and subtask to cluster utterances, since the language generator would typically be provided with this information in a dialog system.

For the three models utterance-string, template-string and dialogact*subtask we approximate equation 1 as equation 2. We then approximate this by selecting the best cluster C^* and searching for the member in the cluster that maximizes $P(W|C^*, H)$ (equation 3). We estimate $P(C|H)$ and $P(W|C^*, H)$ using maximum entropy models. For $P(C|H)$ we train a one-vs-other binary classifier with the features listed in Table 1 and using the LLAMA toolkit [19]. For $P(W|C^*, H)$, we use the same procedure but add the predicted cluster as a feature.

$$= \underset{W}{argmax} \sum_{C} P(C|H) * P(W|C, H) \quad (2)$$

$$\approx \underset{W}{argmax}\ P(W|C^*, H) \quad (3)$$

$$where\ C^* = \underset{C}{argmax}\ P(C|H) \quad (4)$$

For models utterance-bag and template-bag, we compute $P(C|H)$ as described above. Then we use an utterance reconstruction model as specified in equation 6. The tokens in the identified cluster are permuted ($\Pi(C^*)$) and the likelihood for each permutation is computed using an n-gram model. The string with the highest likelihood is returned as the result of generation.

$$W^* = \underset{W \in \Pi(C^*)}{argmax}\ P(W) \quad (5)$$

$$= \underset{w_1 w_2 \ldots w_{|C^*|} \in \Pi(C^*)}{argmax} \prod_{i=1}^{|C^*|} P(w_i|w_{i-1}, \ldots, w_{i-n+1}) \quad (6)$$

4. DATA

We use two corpora: the publicly available 648-dialog Communicator dialog act tagged corpus, and CHILD, a proprietary corpus of 832 customer service dialogs in a catalog ordering domain. Both corpora contain highly-structured task-oriented dialogs. The Communicator dialogs are human-computer dialogs from nine different systems, while CHILD consists of human-human dialogs with multiple different agents. Communicator is annotated using DATE [20] for task/subtask, dialog acts and named entities, permitting us to construct utterance, template and dialogact*subtask models. CHILD is annotated for task/subtask, permitting us to construct utterance and dialogact*subtask models. In both corpora, we used the human transcriptions of the system's/agent's utterances. For the Communicator corpus, we focus on predicting the system utterances, while for CHILD, we predict the customer service agent's utterances.

We split the Communicator corpus along dialog lines into 80% training data, 10% development data and 10% testing data; we split the CHILD corpus into 10% testing data and 90% training data. Prior to building our models, we preprocess and cluster the system's or agent's utterances. We replace numbers with `[number]` and remove closed-class words (articles, prepositions and conjunctions). For models template-bag, template-string and dialogact*subtask, we replace named entities with their types (e.g. *phoenix* with `[city]`). For models utterance-bag and template-bag, we sort the remaining words in the utterance into alphabetical order. We then cluster the utterances, templates or bags of words using string equality. We also extract the features shown in Table 1 from each corpus (the discourse features are not used in model dialogact*subtask).

5. RESULTS

The performance of our system is shown in Figures 1 and 2 for the Communicator corpus and in Figures 3 and 4 for CHILD. We use classification error rate (1-best classification) to evaluate the performance of the cluster identification task. We use string edit distance to evaluate the model's performance overall[1]. When computing string edit distance, we use as reference string the utterance produced by the system/agent, with numbers replaced by [number].

Preserving the order of words in the cluster members increases the numbers of clusters, making the cluster identification problem slightly harder. However, this does not necessarily make the overall surface realization problem harder; Figure 2 shows that without system utterances as context, performance is best for the template-string model on Communicator, and second-best for utterance-string. Although we could not build template-based models for CHILD (because CHILD is not annotated for domain entities), for this corpus

[1]The string edit distance between two strings has value at most 1 and at least the difference between the lengths of the two strings, so it can be negative.

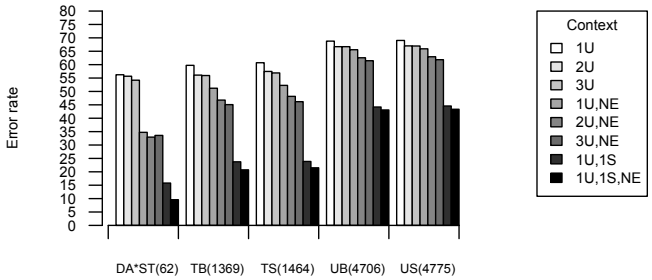

Fig. 1. Classification error rates for cluster prediction (Communicator). xU = features from x previous user turns; xS = features from x previous system turns; NE = named entity features.

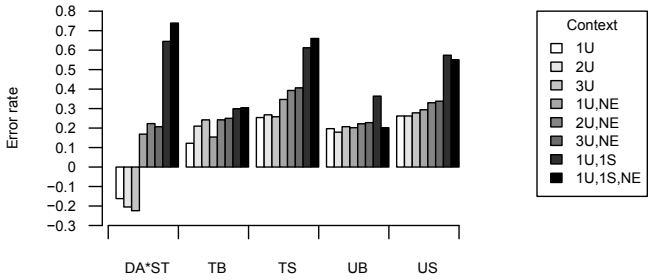

Fig. 2. String edit distance results for generation on Communicator. xU = features from x previous user turns; xS = features from x previous system turns; NE = named entity features.

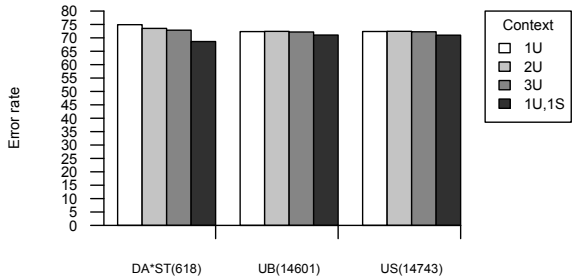

Fig. 3. Classification error rates for cluster prediction on CHILD. xU = features from x previous user turns; xS = features from x previous system turns; NE = named entity features.

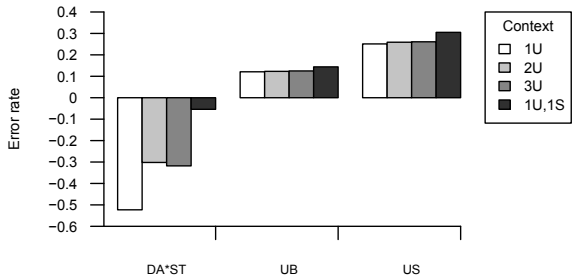

Fig. 4. String edit distance results for generation on CHILD. xU = features from x previous user turns; xS = features from x previous system turns; NE = named entity features.

utterance-based models perform best. This is because the number of utterances per cluster is quite small for template- and utterance-based models compared to dialogact*subtask models, while in the template-based models named entities are not included in the templates, further reducing the number of items per cluster.

Increasing the number of previous user utterances included in the history decreases the classification error rate. The biggest reduction in error rate occurs when the previous system turn is taken as a feature. For these experiments, we have used the correct previous system utterance as a feature as opposed to the previous predicted system utterance. However, when decoding in a dialog system, a dynamic programming approach would be required to get the system's previous utterance, which would affect performance.

Most of the clusters produced by the simple methods we use here contain very similar sentences. For example, in the template-bag model the cluster with words `day like travel what would you` contains utterances like *and what day would you like to travel on* and *and what day would you like to travel*. However, the clusters produced for model dialog act*subtask for Communicator contain sentences with quite different structure and semantics. For example, the cluster `request_info x depart_arrive_date` contains the sentence *and what day would you like to travel on*, as well as sentences like *and departing [city] on what day* and *and on what date didja wanna fly*, more general sentences like *and what month was that*, and error-handling utterances like *but i didn't catch the date could you repeat it please* and *you must specify a date*. Interestingly, the clusters for model dialogact*subtask for CHILD contain utterances that are much more similar in meaning, because CHILD is labeled with a more fine-grained set of dialog act tags. For example, there are separate clusters for `Request (Address) x contact-info` and `Request (Address) x shipping -address`. This also means that there are about 10 times as many clusters in this model for CHILD than for Communicator. Ideally, we would like a cluster construction method that achieves maximum separability in the partitioning of utterances into clusters.

There are two ways generation errors can occur in our system: the wrong cluster can be chosen, or a string can be chosen from the cluster that is not similar to the reference string. Because of the nature of our clusters, the utterances produced by the -string and -bag models generally have very similar import to the reference utterances. For example, these utterances from CHILD are similar in meaning although different in wording: *your phone number with the area code [number]* and *may i have your home telephone number with area code please*. This tends to be true when comparing generated and reference strings from the Communicator corpus even if they belong to different clusters and/or score low using simple string accuracy. For example, both *hello* and *hi welcome to SRI's communicator demonstration* are greetings, but they belong to different template clusters in the Communicator template-string model. However, for CHILD it is not as true that sentences from different clusters tend to have similar meanings. It is unfortunately impossible when string edit distance alone to adequately account for meaning similarity between a pair of utterances. In future work, we will examine improved evaluation metrics for the generation task.

Sometimes, the generated sentence seems more helpful or less awkward than the reference sentence. For example, compare *from which city would you like to fly to salt lake city* (reference) to *where are you departing from* (generated). However, the utterances generated by the dialogact*subtask model frequently could not replace the reference utterances even if they come from the same cluster. For example, from Communicator compare *what time would you like to leave on tuesday october seventeenth* (reference) to *what time do you want to leave anchorage* (generated); and from CHILD compare *and i thank you very much* (reference) to *when you pick up your order you will be asked to show the credit card used for payment* (generated).

Communicator is a corpus of human-computer dialogs; we have shown that we can, in a sense, reproduce a template-based surface realizer from data and use data from multiple systems to introduce more variety into a template-based surface realizer. However, a much more interesting question is whether we can induce a surface realizer from human-human dialogs. Figures 3 and 4 show small differences between performance of the utterance-based models for CHILD. We think this is due to data sparsity – a large number of clusters with few elements in each cluster, and fewer utterances in both test and training data. If we build template-based models in this domain, we expect that we will see a big improvement in performance.

6. CONCLUSIONS

In this paper, we propose a statistical two-stage approach to response generation for dialog. The first stage is mainly concerned with word choice, and the second stage with word order. Both stages can be trained from spoken dialog corpora. We have conducted an evaluation of our model using human-human and human-computer dialogs, and shown that our approach is feasible for word choice.

We are currently exploring additional features, improved models, and improved evaluation metrics for this task. We plan to repeat our experiments with domain entity features after annotating CHILD for domain entities, as well as adding syntactic features such as tense and verb type. We are planning to explore more complex clustering mechanisms for utterances, including sentence simplification, partial parsing and LSA. Finally, because no existing automatic evaluation metric for surface realization performs well in the presence of variation in lexical choice [21, 22], we plan a human evaluation to better measure where our models are succeeding and failing.

7. REFERENCES

[1] S. Channarukul, S. McRoy, and S. Ali, "YAG: A template-based text realization system for dialog," *The International Journal of Uncertainty, Fuzziness, and Knowledge-based Systems*, vol. 9, no. 6, 2001.

[2] M. Johnston et al., "MATCH: An architecture for multimodal dialogue systems," in *Proceedings of ACL 2001*, 2001.

[3] A. Stent, "Content planning and generation in continuous-speech spoken dialog systems," in *Proceedings of the KI'99 Workshop "May I Speak Freely"*, 1999.

[4] M. Walker et al., "Generation and evaluation of user tailored responses in multimodal dialogue," *Cognitive Science*, vol. 28, 2004.

[5] J. Chen, S. Bangalore, O. Rambow, and M. Walker, "Towards automatic generation of natural language generation systems," in *Proceedings of COLING 2002*, 2002.

[6] O. Rambow, S. Bangalore, and M. Walker, "Natural language generation in dialog systems," in *Proceedings of HLT 2001*, 2001.

[7] M. Walker, A. Stent, F. Mairesse, and R. Prasad, "Individual and domain adaptation in sentence planning for dialogue," *Journal of Artificial Intelligence Research*, To appear.

[8] I. Langkilde-Geary, "An empirical verification of coverage and correctness for a general-purpose sentence generator," in *Proceedings of INLG 2002*, 2002.

[9] N. Chambers, "Real-time stochastic language generation for dialogue systems," in *Proceedings of ENLG 2005*, 2005.

[10] A. Oh and A. Rudnicky, "Stochastic language generation for spoken dialogue systems," in *Proceedings of the ANLP/NAACL Workshop on Conversational Systems*, 2000.

[11] S. Corston-Oliver et al., "An overview of Amalgam: A machine-learned generation module," in *Proceedings of INLG 2002*, 2002.

[12] T. Marciniak and M. Strube, "Classification-based generation using TAG," in *Proceedings of INLG 2004*, 2004.

[13] S. Bangalore and O. Rambow, "Exploiting a probabilistic hierarchical model for generation," in *Proceedings of COLING 2000*, 2000.

[14] H. Zhong and A. Stent, "Building surface realizers automatically from corpora," in *Proceedings of the 2005 Workshop on Using Corpora in Natural Language Generation*, 2005.

[15] A. Belz, "Probabilistic generation of weather forecast texts," in *Proceedings of NAACL HLT 2007*, 2007.

[16] I. Langkilde-Geary, "An exploratory application of constraint optimization in Mozart to probabilistic natural language processing," in *Proceedings of CSLP 2004*, 2004.

[17] C. Lewis and G. Di Fabbrizio, "Prompt selection with reinforcement learning in an AT&T call routing application," in *Proceedings of ICSLP 2006*, 2006.

[18] R. Higashinaka, M. Walker, and R. Prasad, "Learning to generate naturalistic utterances using reviews in spoken dialogue systems," *ACM Transactions on Speech and Language Processing*, vol. In press, 2007.

[19] P. Haffner, "Scaling large margin classifiers for spoken language understanding," *Speech Communication*, vol. 48, no. iv, 2006.

[20] W. Hastie, H. Prasad, and R. Walker, "Automatic evaluation: Using a date dialogue act tagger for user satisfaction and task completion prediction," in *In Proc. of LREC*, 2002.

[21] A. Belz and E. Reiter, "Comparing automatic and human evaluation of NLG systems," in *Proceedings of EACL 2006*, 2006.

[22] A. Stent, M. Marge, and M. Singhai, "Evaluating evaluation methods for generation in the presence of variation," in *Proceedings of CICLing 2005*, 2005.