Towards Automatic Generation of Natural Language Generation Systems

John Chen*, **Srinivas Bangalore**†, **Owen Rambow***, and **Marilyn A. Walker**†
Columbia University* AT&T Labs–Research†
New York, NY 10027 Florham Park, NJ 07932
{jchen,rambow}@cs.columbia.edu {srini,walker}@research.att.com

Abstract

Systems that interact with the user via natural language are in their infancy. As these systems mature and become more complex, it would be desirable for a system developer if there were an automatic method for creating natural language generation components that can produce quality output efficiently. We conduct experiments that show that this goal appears to be realizable. In particular we discuss a natural language generation system that is composed of SPoT, a trainable sentence planner, and FERGUS, a stochastic surface realizer. We show how these stochastic NLG components can be made to work together, that they can be ported to new domains with apparent ease, and that such NLG components can be integrated in a real-time dialog system.

1 Introduction

Systems that interact with the user via natural language are in their infancy. As these systems mature and become more complex, it would be desirable for a system developer if there were automatic methods for creating natural language generation (NLG) components that can produce quality output efficiently. Stochastic methods for NLG may provide such automaticity, but most previous work (Knight and Hatzivassiloglou, 1995), (Langkilde and Knight, 1998), (Oh and Rudnicky, 2000), (Uchimoto et al., 2000), (Bangalore and Rambow, 2000) concentrate on the specifics of individual stochastic methods, ignoring other issues such as integrability, portability, and efficiency. In contrast, this paper investigates how different stochastic NLG components can be made to work together effectively, whether they can easily be ported to new domains, and whether they can be integrated in a real-time dialog system.

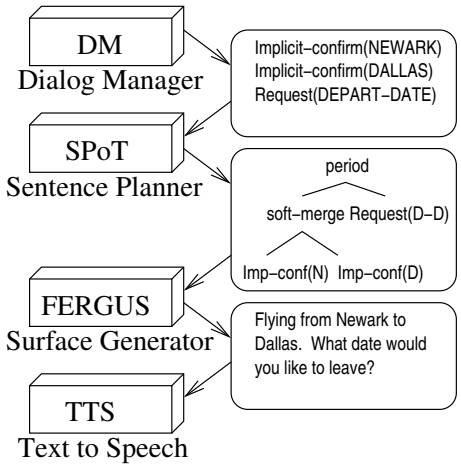

Figure 1: Components of an NLG system.

Recall the basic tasks in NLG. During text planning, content and structure of the target text are determined to achieve the overall communicative goal. During sentence planning, linguistic means—in particular, lexical and syntactic means—are determined to convey smaller pieces of meaning. During realization, the specification chosen in sentence planning is transformed into a surface string by linearizing and inflecting words in the sentence (and typically, adding function words). Figure 1 shows how such components cooperate to generate text corresponding to a set of communicative goals.

Our work addresses both the sentence planning stage and the realization stage. The sentence planning stage is embodied by the SPoT sentence planner (Walker et al., 2001), while the surface realization stage is embodied by the FERGUS surface realizer (Bangalore and Rambow, 2000). We extend the work of (Walker et al., 2001) and (Bangalore and Rambow, 2000) in various ways. We show that apparently each

of SPoT and FERGUS can be ported to different domains with little manual effort. We then show that these two components can work together effectively. Finally, we show the on-line integration of FERGUS with a dialog system.

2 Testing the Domain Independence of Sentence Planning

In this section, we address the issue of the amount of effort that is required to port a sentence planner to new domains. In particular, we focus on the SPoT sentence planner. The flexibility of the training mechanism that SPoT employs allows us to perform experiments that provide evidence for its domain independence.

Being a sentence planner, SPoT chooses abstract linguistic resources (meaning-bearing lexemes, syntactic constructions) for a text plan. A text plan is a set of communicative goals which is assumed to be output by a dialog manager of a spoken dialog system. The output of SPoT is a set of ranked sentence plans, each of which is a binary tree with leaves labeled by the communicative goals of the text plan.

SPoT divides the sentence planning task into two stages. First, the sentence-plan-generator (SPG) generates 12-20 possible sentence plans for a given input text plan. These are generated randomly by incrementally building each sentence plan according to some probability distribution. Second, the sentence-plan-ranker (SPR) ranks the resulting set of sentence plans. SPR is trained for this task via RankBoost (Freund et al., 1998), a machine learning algorithm, using as training data sets of sentence plans ranked by human judges.

In porting SPoT to a new domain, this last point seems to be a hindrance. New training data in the new domain ranked by human judges might be needed in order to train SPoT. To the contrary, our experiments that show that this need not be the case. We partition the set of all features used by (Walker et al., 2001) to train SPoT into three subsets according to their level of domain and task dependence. Domain independent features are features whose names include only closed-class words, e.g. "in," or names of operations that incrementally build the sentence plan, e.g. *merge*. Domain-dependent, task-independent features are those whose names include open class words specific to this domain, e.g. "travel" or the names of the role slots, e.g. `$DEST-CITY`. Domain dependent, task dependent features are features whose names include the value of a role filler for the domain, e.g. "Albuquerque."

Features Used	Mean Score	S.D.
ALL	4.56	0.68
DOMAIN-INDEPENDENT	4.55	0.69
TASK-INDEPENDENT	4.20	0.99
TASK-DEPENDENT	3.90	1.19

Table 1: Results for subsets of features used to train SPoT

We have trained and tested SPoT with these different feature subsets using the air-travel domain corpus of 100 text plans borrowed from (Walker et al., 2001), using five fold cross-validation. Results are shown in Table 2 using t-tests with the modified Bonferroni statistic for multiple comparisons. Scores can range from 1.0 (worst) to 5.0 (best). The results indicate that the domain independent feature set performs as well as all the features ($t = .168$, $p = .87$), but that both the task independent ($t = 6.25$, $p = 0.0$) and the task dependent ($t = 4.58$, $p = 0.0$) feature sets perform worse.

3 Automation in Training a Surface Realizer

As with the sentence planning task, there is the possibility that the task of surface realization may be made to work in different domains with relatively little manual effort. Here, we perform experiments using the FERGUS surface realizer to determine whether this may be so. We review the FERGUS architecture, enumerate resources required to train FERGUS, recapitulate previous experiments that indicate how these resources can be automatically generated, and finally show how similar ideas can be used to port FERGUS to different domains with little manual effort.

3.1 Description of the FERGUS Surface Realizer

Given an underspecified dependency tree representing one sentence as input, FERGUS outputs the best surface string according to its stochastic modeling. Each node in the input tree corresponds to a lexeme. Nodes that are related by

grammatical function are linked together. Surface ordering of the lexemes remains unspecified in the tree.

FERGUS consists of three models: tree chooser, unraveler, and linear precedence chooser. The tree chooser associates a supertag (Bangalore and Joshi, 1999) from a tree-adjoining grammar (TAG) with each node in the underspecified dependency tree. This partially specifies the output string's surface order; it is constrained by grammatical constraints encoded by the supertags (e.g. subcategorization constraints, voice), but remains free otherwise (e.g. ordering of modifiers). The tree chooser uses a stochastic tree model (TM) to select a supertag for each node in the tree based on local tree context. The unraveler takes the resulting semi-specified TAG derivation tree and creates a word lattice corresponding to all of the potential surface orderings consistent with this tree. Finally, the linear precedence (LP) chooser finds the best path through the word lattice according to a trigram language model (LM), specifying the output string completely.

Certain resources are required in order to train FERGUS. A TAG grammar is needed—the source of the supertags with which the semi-specified TAG derivation tree is annotated. There needs to be a treebank in order to obtain the stochastic model TM driving the tree chooser. There also needs to be a corpus of sentences in order to train the language model LM required for the LP chooser.

3.2 Labor-Minimizing Approaches to Training FERGUS

The resources that are needed to train FERGUS seem quite labor intensive to develop. But (Bangalore et al., 2001) show that automatically generated version of these resources can be used by FERGUS to obtain quality output.

Two kinds of TAG grammar are used in (Bangalore et al., 2001). One kind is a manually developed, broad-coverage grammar for English: the XTAG grammar (XTAG-Group, 2001). It consists of approximately 1000 tree frames. Disadvantages of using XTAG are the considerable amount of human labor expended in its development and the lack of a treebank based on XTAG—the only way to estimate parameters in the TM is to rely on a heuristic mapping of XTAG tree frames onto a pre-existing treebank (Bangalore and Joshi, 1999). Another kind of grammar is a TAG automatically extracted from a treebank using the techniques of (Chen, 2001) (cf. (Chiang, 2000), (Xia, 1999)). These techniques extract a linguistically motivated TAG using heuristics programmed using a modicum of human labor. They nullify the disadvantages of using the XTAG grammar, but they introduce potential complications—notably, an extracted grammar's size is often much larger than that of XTAG, typically more than 2000 tree frames, potentially leading to a larger sparse data problem, and also the resulting grammar is not hand-checked.

Two kinds of treebank are used in (Bangalore et al., 2001). One kind is the Penn Treebank (Marcus et al., 1993). It consists of approximately 1,000,000 words of hand-checked, bracketed text. The text consists of Wall Street Journal news articles. The other kind of treebank is the BLLIP corpus (Charniak, 2000). It consists of approximately 40,000,000 words of text that has been parsed by a broad-coverage statistical parser. The text consists of Wall Street Journal news and newswire articles. The advantage of the former is that it has been hand-checked, whereas the latter has the advantage of being easily produced and hence can easily be enlarged.

(Bangalore et al., 2001) experimentally determine how the quality and quantity of the resources used in training FERGUS affect the output quality of the generator. They find that while a better quality annotated corpus (Penn Treebank) results in better model accuracy than a lower quality corpus (BLLIP) of the same size, an (easily-obtained) larger lower quality corpus results in a model that eclipses a smaller, better quality treebank. Also, the model that is obtained when using an automatically extracted grammar yields comparable output quality to the model that is obtained when using a hand-crafted (XTAG) grammar.

3.3 Automating Adaptation of FERGUS to a New Domain

This paper is about minimizing the amount of manual labor that is required to port NLG components to different domains. (Bangalore et al., 2001) perform all of their experiments on the same domain of Wall Street Journal news articles. In contrast, in this section we show

that FERGUS can be adapted to the domain of air-travel reservation dialogs with minimal human effort. We show that out-of-domain training data can be used instead of in-domain data without drastically compromising output quality. We also show that partially parsed in-domain training data can be effectively used to train the TM. Finally, we show that using an in-domain corpus to train the LM can help the output quality, even if that corpus is of small size. In this section, we first describe the training resources that are used in these experiments. We subsequently describe the experiments themselves and their results.

Various corpora are used in these experiments. For training, there are two distinct corpora. First, there is the previously introduced Penn Treebank (PTB). As the alternative, there is a human-human corpus of dialogs (HH) from Carnegie Mellon University. The HH corpus consists of approximately 13,000 words in the air-travel reservation domain. This is not exactly the target domain because human-human interaction differs from human-computer interaction which is our true target domain. From this raw text, an LDA parser (Bangalore and Joshi, 1999) trained using the XTAG-based Penn Treebank creates a partially-parsed, non-hand-checked treebank. Test data consists of about 2,200 words derived from Communicator template data. Communicator templates are hand-crafted surface strings of words interspersed with slot names. An example is "`What time would you, traveling from $ORIG-CITY to $DEST-CITY like to leave?`" The test data is derived from all strings like these, with duplicates, in the Communicator system by replacing the slot names with fillers according to a probability distribution. Furthermore, dependency parses are assigned to the resulting strings by hand.

In the first series of experiments, we ascertain the output quality of FERGUS using the XTAG grammar on different training corpora. We vary the TM's training corpus to be either PTB or HH. We do the same for the LM's training corpus. Assessing the output quality of a generator is a complex issue. Here, we select as our metric *understandability accuracy*, defined in (Bangalore et al., 2000) as quantifying the difference between the generator output, in terms of both dependency tree and surface string, and the desired reference output. (Bangalore et al., 2000) finds this metric to correlate well with human judgments of understandability and quality. Understandability accuracy varies between a high score of 1.0 and a low score which may be less than zero.

	PTB TM	HH TM
PTB LM	0.30	0.38
HH LM	0.37	0.41

Table 2: Average understandability accuracies using XTAG-Based FERGUS for various kinds of training data

	PTB TM
PTB LM	0.39
HH LM	0.33

Table 3: Average understandability accuracies using automatically-extracted grammar based FERGUS for various kinds of training data

The results of our experiments are shown in Table 2. We conclude that despite its smaller size, and despite its being only automatically- and partially- parsed, using the in-domain HH is more effective than using the out-of-domain PTB for training the TM. Similarly, HH is more effective than PTB for training the LM. The best result is obtained by using HH to train both the TM and the LM; this result (0.41) is comparable to the result obtained by using matched PTB training and test data (0.43) that is used in (Bangalore et al., 2001).

The second series of experiments investigates the output quality of FERGUS using automatically-extracted grammars. In these experiments, the TM is always trained on PTB but not HH. It is the type of training data that is used to train the LM, either PTB or HH, that is varied. The results are shown in Table 3. Note that these scores are in the same range as those obtained when training FERGUS using XTAG. Also, these scores show that when using automatically-extracted grammars, training LM using a large, out-of-domain corpus (PTB) is more beneficial than training LM using a small, in-domain corpus (HH).

We can now draw various conclusions about training FERGUS in a new domain. Consider training the TM. It is not necessary to use a handwritten TAG in the new domain; a broad-coverage hand-written TAG or an automatically-extracted TAG will give comparable results. Also, instead of requiring a hand-checked treebank in the new domain, partially parsed data in the new domain is adequate. Now consider training the LM. Our experiments show that a small corpus in the new domain is a viable alternative to a large corpus that is out of the domain.

4 Integration of SPoT with FERGUS

We have seen evidence that both SPoT and FERGUS may be easily transferable to a new domain. Because the output of a sentence planner usually becomes the input of a surface realizer, questions arise such as whether SPoT and FERGUS can be made to work together in a new domain and what is the output quality of the combined system. We will see that an addition of a rule-based component to FERGUS will be necessary in order for this integration to occur. We will subsequently see that the output quality of the resulting combination of SPoT and FERGUS is quite good.

Integration of SPoT as described in Section 2 and FERGUS as described in Section 3 is not automatic. The reason is that the output of SPoT is a deep syntax tree (Mel'čuk, 1998) whereas hitherto the input of FERGUS has been a surface syntax tree. The primary distinguishing characteristic of a deep syntax tree is that it contains features for categories such as definiteness for nouns, or tense and aspect for verbs. In contrast, a surface syntax tree realizes these features as function words. However, there is a one-to-one mapping from features of a deep syntax tree to function words in the corresponding surface syntax tree. Therefore, integrating SPoT with FERGUS is basically a matter of performing this mapping. We have added a rule-based component (RB) as the new first stage of FERGUS to do just that. Note that it is erroneous to think that RB makes choices between different generation options because there is a one-to-one mapping between features and function words.

	PTB TM	HH TM
PTB LM	0.48	0.47
HH LM	0.73	0.68

Table 4: Average understandability accuracies of SPoT-integrated, XTAG-Based FERGUS for various kinds of training data

After the addition of RB to FERGUS, we evaluate the output quality of the combination of SPoT and FERGUS. Only the XTAG grammar is used in this experiment. As in previous experiments with the XTAG grammar, there is either the option of training using HH or PTB derived data for either the TM or LM, giving a total of four possibilities.

Test data is obtained by output strings that are produced by the combination of SPoT and the RealPro surface realizer (Lavoie and Rambow, 1998). RealPro has the advantage of producing high quality surface strings, but at the cost of having to be hand-tuned to a particular domain. It is this cost we are attempting to minimize by using FERGUS. Only those sentence plans produced by SPoT ranked 3.0 or greater by human judges are used. The surface realization of these sentence plans yields a test corpus of 2,200 words.

As shown in Table 4, the performance of SPoT and FERGUS combined is quite high. Also note that in terms of training the LM, output quality is markedly better when HH is used rather than PTB. Furthermore, note that there is a smaller difference between using PTB or HH to train TM when compared to previous results shown in Table 2. This seems to indicate that the TM's effect on output quality diminishes because of addition of RB to FERGUS.

5 On-line Integration of FERGUS with a Dialog System

Certain statistical natural language processing systems can be quite slow, usually because of the large search space that these systems must explore. It is therefore uncertain whether a statistical NLG component can be integrated into a real-time dialog system. Investigating the matter in FERGUS's case, we have experimented with integrating FERGUS into Communicator, a mixed-initiative, airline travel reservation sys-

tem. We begin by explaining how Communicator manages surface generation without FERGUS. We then delineate several possible kinds of integration. Finally, we describe our experiences with one kind of integration.

Communicator performs only a rudimentary form of surface generation as follows. The dialog manager of Communicator issues a set of communicative goals that are to be realized. Surface template strings are selected based on this set, such as "`What time would you, traveling from $ORIG-CITY to $DEST-CITY like to leave?`" The slot names in these strings are then replaced with fillers according to the dialog manager's state. The resulting strings are then piped to a text-to-speech component (TTS) for output.

There are several possibilities as to how FERGUS may supplant this system. One possibility is off-line integration. In this case, the set of all possible sets communicative goals for which the dialog manager requires realization are matched with a set of corresponding surface syntax trees. The latter set is input to FERGUS, which generates a set of surface template strings, which in turn is used to replace the manually created surface template strings that are an original part of Communicator. Since these changes are precompiled, the resulting version of Communicator is therefore as fast the original. On the other hand, off-line integration may be unmanageable if the set of sets of communicative goals is very large. In that case, only the alternative of on-line integration is palatable. In this approach, each surface template string in Communicator is replaced with its corresponding surface syntax tree. At points in a dialog where Communicator requires surface generation, it sends the appropriate surface syntax trees to FERGUS, which generates surface strings.

We have implemented the on-line integration of FERGUS with Communicator. Our experiments show that FERGUS is fast enough to be used in for this purpose, the average time for FERGUS to generate output strings for one dialog turn being only 0.28 seconds.

6 Conclusions and Future Work

We have performed experiments that provide evidence that components of a statistical NLG system may be ported to different domains without a huge investment in manual labor. These components include a sentence planner, SPoT, and a surface realizer, FERGUS. SPoT seems easily portable to different domains because it can be trained well using only domain-independent features. FERGUS may also be said to be easily portable because our experiments show that the quality and quantity of in-domain training data need not be high and plentiful for decent results. Even if in-domain data is not available, we show that out-of-domain training data can be used with adequate results.

By integrating SPoT with FERGUS, we have also shown that different statistical NLG components can be made to work well together. Integration was achieved by adding a rule-based component to FERGUS which transforms deep syntax trees into surface syntax trees. The combination of SPoT and FERGUS performs with high accuracy. Post-integration, there is a diminishing effect of TM on output quality.

Finally, we have shown that a statistical NLG component can be integrated into a dialog system in real time. In particular, we replace the hand-crafted surface generation of Communicator with FERGUS. We show that the resulting system performs with low latency.

This work may be extended in different directions. Our experiments showed promising results in porting to the domain of air travel reservations. Although this is a reasonably-sized domain, it would be interesting to see how our findings vary for broader domains. Our experiments used a partially parsed version of the HH corpus. We would like to compare its use as TM training data in relation to using a fully parsed version of HH, and also a hand-checked treebank version of HH. We would also like to investigate the possibility of interpolating models based on different kinds of training data in order to ameliorate data sparseness. Our experiments focused on integration between the NLG components of sentence planning and surface generation. We would like to explore the possibility of further integration, in particular integrating these components with TTS. This would provide the benefit of enabling the use of syntactic and semantic information for prosody assignment. Also, although FERGUS was integrated with SPoT relatively easily, it does not necessarily follow that FERGUS can be inte-

grated easily with other kinds of components. It may be worthwhile to envision a redesigned version of FERGUS whose input can be flexibly underspecified in order to accommodate different kinds of modules.

7 Acknowledgments

This work was partially funded by DARPA under contract MDA972-99-3-0003.

References

Srinivas Bangalore and A. K. Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2).

Srinivas Bangalore and Owen Rambow. 2000. Exploiting a probabilistic hierarchical model for generation. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*.

Srinivas Bangalore, Owen Rambow, and Steve Whittaker. 2000. Evaluation metrics for generation. In *Proceedings of the First International Conference on Natural Language Generation*, Mitzpe Ramon, Israel.

Srinivas Bangalore, John Chen, and Owen Rambow. 2001. Impact of quality and quantity of corpora on stochastic generation. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Langauge Processing*, Pittsburgh, PA.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of First Annual Meeting of the North American Chapter of the Association for Computational Linguistics*, Seattle, WA.

John Chen. 2001. *Towards Efficient Statistical Parsing Using Lexicalized Grammatical Information*. Ph.D. thesis, University of Delaware.

David Chiang. 2000. Statistical parsing with an automatically-extracted tree adjoining grammar. In *Proceedings of the the 38th Annual Meeting of the Association for Computational Linguistics*, pages 456–463, Hong Kong.

Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. 1998. An efficient boosting algorithm for combining preferences. In *Machine Learning: Proceedings of the Fifteenth International Conferece*.

Kevin Knight and V. Hatzivassiloglou. 1995. Two-level many-paths generation. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, Boston, MA.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics*, Montreal, Canada.

Benoit Lavoie and Owen Rambow. 1998. A framework for customizable generation of multi-modal presentations. In *Proceedings of the 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics*, Montreal, Canada.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, 19(2):313–330.

Igor A. Mel'čuk. 1998. *Dependency Syntax: Theory and Practice*. State University of New York Press, New York, NY.

Alice H. Oh and Alexander I. Rudnicky. 2000. Stochastic language generation for spoken dialog systems. In *Proceedings of the ANLP/NAACL 2000 Workshop on Conversational Systems*, pages 27–32, Seattle, WA.

Kiyotaka Uchimoto, Masaki Murata, Qing Ma, Satoshi Sekine, and Hitoshi Isahara. 2000. Word order acquisition from corpora. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING '00)*, Saarbrücken, Germany.

Marilyn A. Walker, Owen Rambow, and Monica Rogati. 2001. Spot: A trainable sentence planner. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 17–24.

Fei Xia. 1999. Extracting tree adjoining grammars from bracketed corpora. In *Fifth Natural Language Processing Pacific Rim Symposium (NLPRS-99)*, Beijing, China.

The XTAG-Group. 2001. A Lexicalized Tree Adjoining Grammar for English. Technical report, University of Pennsylvania. Updated version available at http://www.cis.upenn.edu/~xtag.