Practical Grammar-Based NLG from Examples

David DeVault and **David Traum** and **Ron Artstein**
USC Institute for Creative Technologies
13274 Fiji Way
Marina del Rey, CA 90292
{devault,traum,artstein}@ict.usc.edu

Abstract

We present a technique that opens up grammar-based generation to a wider range of practical applications by dramatically reducing the development costs and linguistic expertise that are required. Our method infers the grammatical resources needed for generation from a set of declarative examples that link surface expressions directly to the application's available semantic representations. The same examples further serve to optimize a run-time search strategy that generates the best output that can be found within an application-specific time frame. Our method offers substantially lower development costs than hand-crafted grammars for application-specific NLG, while maintaining high output quality and diversity.

1 Introduction

This paper presents a new example-based generation technique designed to reduce the development costs and linguistic expertise needed to integrate a grammar-based generation component into an existing application. We believe this approach will broaden the class of applications in which grammar-based generation may feasibly be deployed.

In principle, grammar-based generation offers significant advantages for many applications, when compared with simpler template-based or canned text output solutions, by providing productive coverage and greater output variety. However, realizing these advantages can require significant development costs (Busemann and Horacek, 1998).

One possible strategy is to exploit a wide-coverage realizer that aims for applicability in multiple application domains (White et al., 2007; Cahill and van Genabith, 2006; Zhong and Stent, 2005; Langkilde-Geary, 2002; Langkilde and Knight, 1998; Elhadad, 1991). These realizers provide a sound wide-coverage grammar (or robust wide-coverage language model) for free, but demand a specific input format that is otherwise foreign to an existing application. Unfortunately, the development burden of implementing the translation between the system's available semantic representations and the required input format can be quite substantial (Busemann and Horacek, 1998). Indeed, implementing the translation might require as much effort as would be required to build a simple custom generator; cf. (Callaway, 2003). Thus, there currently are many applications where using a wide-coverage generator remains impractical.

Another strategy is for system builders to hand craft an application-specific grammar for generation. This approach can be initially attractive to system builders because it allows syntactic coverage and semantic modeling to be tailored directly to application needs. However, writing grammatical rules by hand ultimately requires a painstaking, time-consuming effort by a developer who has detailed linguistic knowledge as well as detailed application knowledge. Further, the resulting coverage is inevitably limited to the set of linguistic constructions that have been selected for careful modeling.

A third strategy is to use an example-based approach (Wong and Mooney, 2007; Stone, 2003; Varges and Mellish, 2001) in which the connection between available application semantic representations and desired output utterances is specified by example. Example-based approaches aim to allow system builders to specify a productive generation capacity while leaving the representations and reasoning that underlie that productive capacity mostly implicit in a set of training examples. This methodology insulates system builders from the detailed expertise and technical infrastructure needed to implement the productive capacity directly, and has made example-based approaches attractive not only in text generation but also in related areas such as concatenative speech synthesis and motion capture based animation; see, e.g., (Stone et al., 2004).

The technique we present in this paper is a new example-based approach to specifying application-specific text generation. As in other hand-crafted and example-based approaches, our technique allows syntactic coverage and semantic modeling to follow the needs and available semantic representations in an application. One contribution of our technique is to relieve the generation content author of the burden of manual syntactic modeling by leveraging an off-the-shelf parser; defects in the syntax provided by the parser are effectively overcome using a machine learning technique. Additionally, our technique organizes the authoring task in a way that relieves the generation author of carefully modeling the connections between particular syntactic constructions and available semantic representations.

Together, we argue, these features dramatically reduce the linguistic expertise and other development costs that are required to integrate a grammar-based generation component into an existing system. In a case study application, we show that our approach allows an application developer who lacks detailed linguistic knowledge to extend grammatical coverage at an expense of less than one minute per additional lexical entry.

2 Case Study: Doctor Perez

Our approach has been tested as a replacement for the generation component of interactive virtual humans used for social training purposes (Swartout et al., 2006). Virtual humans are embodied conversational agents that play the role of people in simulations or games. The case study we present in this paper is the generation of output utterances for a particular virtual human, Doctor Perez, who is designed to teach negotiation skills in a multi-modal, multi-party, non-team dialogue setting (Traum et al., 2008). The human trainee who talks to the doctor plays the role of a U.S. Army captain named Captain Kirk. The design goals for Doctor Perez create a number of requirements for a practical NLG component. We briefly summarize these requirements here; see (DeVault et al., 2008) for more details.

Doctor Perez has a relatively rich internal mental state including beliefs, goals, plans, and emotions. He uses an attribute-value matrix (AVM) semantic representation to describe an utterance as a set of core speech acts and other dialogue acts. Speech acts generally have semantic contents that describe propositions and questions about states and actions in the domain. To facilitate interprocess communication, and statistical processing, this AVM structure is linearized into a “frame” of key values in which each non-recursive terminal value is paired with a path from the root to the final attribute. Figure 1 shows a typical frame. See (Traum, 2003) for additional details and examples of this representation.

While only hundreds of frames currently arise in actual dialogues, the number of potential frames is orders of magnitude larger, and it is difficult to predict in advance which frames might occur. The utterances that realize these frames need to take a variety of syntactic forms, including simple declarative sentences, various modal constructions relating to hypothetical actions or plans, yes/no and wh-questions, and abbreviated dialogue forms such as elliptical clarification and repair requests, grounding, and turn-taking utterances. Highly fluent output is not a necessity for this character, since Doctor Perez is designed to simulate a non-native English speaker. However, in order to support compelling real-time conversational interaction and effective training, the generation module must be able to identify an utterance for Doctor Perez to use within approximately 200ms on modern hardware.

Finally, the development team for Doctor Perez’s language capabilities includes approximately 10 programmers, testers, linguists, and computational linguists. Wherever possible, it is better if any developer can improve any aspect of Doctor Perez’s language processing; e.g., if a programmer discovers a bug or disfluency in the NLG output, it is better if she can fix it directly rather than requiring a (computational) linguist to do so.

3 Technical Approach

Our approach builds on recently developed techniques in statistical parsing, lexicalized syntax modeling, generation with lexicalized grammars, and search optimization to automatically construct all the resources needed for a high-quality run-time generation component. In particular, we leverage the increasing availability of off-the-shelf parsers such as (Charniak, 2001; Charniak, 2005) to automatically (or semi-automatically) assign syntactic analyses to a set of suggested output sentences. We then draw on lexicalization techniques for statistical language models (Magerman, 1995; Collins, 1999; Chiang, 2000; Chiang, 2003) to induce a probabilistic, lexicalized tree-adjoining grammar that supports the derivation of all the suggested output sentences, and many others besides.

The final step is to use the training examples to learn an effective search policy so that our run-time generation component can find good output sentences in a reasonable time frame. In particular, we use variants of existing search optimization (Daumé and Marcu, 2005) and ranking algorithms (Collins and Koo, 2005) to train our run-time component to find good outputs within a specified time window; see also (Stent et al., 2004; Walker et al., 2001). The result is a run-time component that treats generation as an anytime search problem, and is thus suitable for applications in which a time/performance tradeoff is necessary (such as real-time dialogue).

3.1 Specification of Training Examples

Each training example in our approach specifies a target output utterance (string), its syntax, and a set of links between substrings within the utterance and system semantic representations. Formally, a training example takes the form $(u, \mathrm{syntax}(u), \mathrm{semantics}(u))$. We will illustrate this format using the training example in Figure 1. In this example, the generation content author suggests the output utterance u = *we don't have medical supplies here captain*. Each utterance u is accompanied by $\mathrm{syntax}(u)$, a syntactic analysis in Penn Treebank format (Marcus et al., 1994). In the figure, we show two alternative syntactic analyses that might be specified: one is the uncorrected output of the Charniak parser on this sentence, and the other a hand-corrected version of that parse; we evaluate the utility of this hand correction in Section 4.

To represent the meaning of utterances, our approach assumes that the system provides some set $M = \{m_1, ..., m_j\}$ of semantic representations. The meaning of any individual utterance is then identified with some subset of M. For Doctor Perez, M comprises the 232 distinct key-value pairs that appear in the system's various generation frames. In this example, the utterance's meaning is captured by the 8 key-value pairs indicated in the figure.

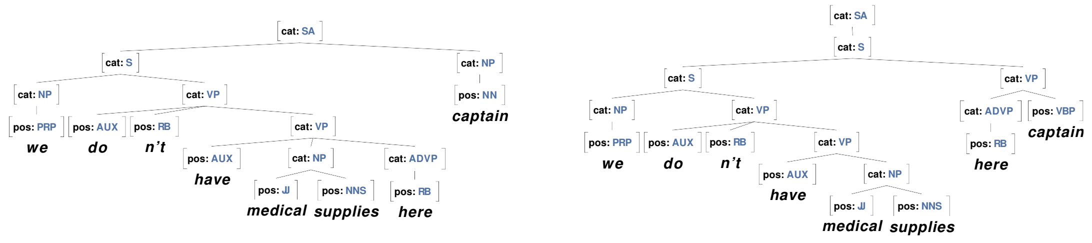
Our approach requires the generation content author to link these 8 key-value pairs to contiguous surface expressions within the utterance. The technique is flexible about which surface expressions are chosen (e.g. they need not correspond to constituent boundaries); however, they do need to be compatible with the way the syntactic analysis tokenizes the utterance, as follows. Let $t(u) = \langle t_1, ..., t_n \rangle$ be the terminals in the syntactic analysis, in left-to-right order. Formally, $\mathrm{semantics}(u) = \{(s_1, M_1), ..., (s_k, M_k)\}$, where $t(u) = s_1 @ \cdots @ s_k$ (with @ denoting concatenation), and where $M_i \subseteq M$ for all $i \in 1..k$. In this example, the surface expression *we don't*, which tokenizes as $\langle \mathrm{we}, \mathrm{do}, \mathrm{n't} \rangle$, is connected to key-values that indicate a negative polarity assertion.

This training example format has two features that are crucial to our approach. First, the semantics of an utterance is specified *independently* of its syntax. This greatly reduces the amount of linguistic expertise a generation content author needs to have. It also allows making changes to the underlying syntax without having to re-author the semantic links.

Second, the assignment of semantic representations to surface expressions must span the *entire utterance*. No words or expressions can be viewed as "meaningless". This is essential because, otherwise, the semantically motivated search algorithm used in generation has no basis on which to include those particular expressions when it constructs its output utterance. Many systems, including Doctor Perez, lack some of the internal representations that would be necessary to specify semantics down to the lex-

Utterance we don't have medical supplies here captain

Syntax



(corrected Charniak parse) *or* (uncorrected Charniak parse)

semantic frame

Semantics		
	we do n't	speech-act.action = assert
		speech-act.content.polarity = negative
	have	speech-act.content.attribute = resourceAttribute
	medical supplies	speech-act.content.value = medical-supplies
	here	speech-act.content.object-id = market
	captain	addressee = captain-kirk
		dialogue-act.addressee = captain-kirk
		speech-act.addressee = captain-kirk

Figure 1: A generation training example for Doctor Perez. If uncorrected syntax is used, the generation content author only writes the utterance and the links to the semantic frame.

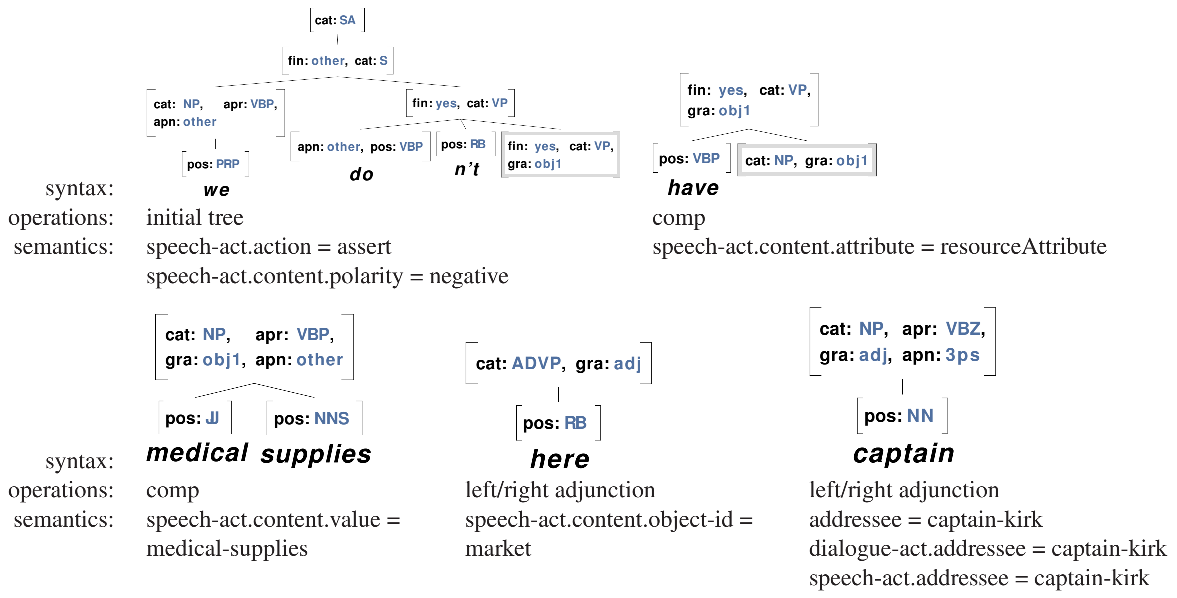
ical level. An important feature of our approach is that it allows an arbitrary semantic granularity to be employed, by mapping the representations available in the system to appropriate multi-word chunks.

3.2 Automatic Grammar Induction

We adopt essentially the probabilistic tree-adjoining grammar (PTAG) formalism and grammar induction technique of (Chiang, 2003). Our approach makes three modifications, however. First, while Chiang's model includes both full adjunction and sister adjunction operations, our grammar has only sister adjunction (left and right), exactly as in the TAGLET grammar formalism of (Stone, 2002). Second, to support lexicalization at an arbitrary granularity, we allow Chiang's tree templates to be associated with more than one lexical anchor. Third, to unify syntactic and semantic reasoning in search, we augment lexical anchors with semantic information. Formally, wherever Chiang's model has a lexical anchor w, ours has a pair $(\langle w_1, ..., w_n\rangle, M')$, where $M' \subseteq M$ is connected to lexical anchors $\langle w_1, ..., w_n\rangle$ by the generation content author, as in Figure 1. The result is that the derivation probabilities the grammar assigns depend not only on the implicated syntactic structures and lexical anchors but also on the *senses* of those lexical anchors in application terms.

We induce our grammar from training examples such as Figure 1 using heuristics to assign derivations to the examples, exactly as in (Chiang, 2003). The process proceeds in two stages. In the first stage, a collection of rules is used to automatically "decorate" the training syntax with a number of features. These include deciding the lexical anchor(s) for each non-terminal constituent and assigning complement/adjunct status for nonterminals which are not on their parent's lexicalization path; see (Magerman, 1995; Chiang, 2003; Collins, 1999). In addition, we deterministically add features to improve several grammatical aspects, including (1) enforcing verb inflectional agreement in derived trees, (2) enforcing consistency in the finiteness of VP and S complements, and (3) restricting subject/direct object/indirect object complements to play the same grammatical role in derived trees.

In the second stage, the complements and adjuncts in the decorated trees are incrementally re-



cat: SA
fin: other, cat: S
cat: NP, apr: VBP, apn: other
pos: PRP
we
fin: yes, cat: VP
apn: other, pos: VBP
do
pos: RB
n't
fin: yes, cat: VP, gra: obj1
fin: yes, cat: VP, gra: obj1
pos: VBP
have
cat: NP, gra: obj1
syntax:
operations:
semantics:
initial tree
speech-act.action = assert
speech-act.content.polarity = negative
comp
speech-act.content.attribute = resourceAttribute
cat: NP, apr: VBP, gra: obj1, apn: other
pos: JJ
pos: NNS
medical supplies
cat: ADVP, gra: adj
pos: RB
here
cat: NP, apr: VBZ, gra: adj, apn: 3ps
pos: NN
captain
syntax:
operations:
semantics:
comp
speech-act.content.value = medical-supplies
left/right adjunction
speech-act.content.object-id = market
left/right adjunction
addressee = captain-kirk
dialogue-act.addressee = captain-kirk
speech-act.addressee = captain-kirk

Figure 2: The linguistic resources inferred from the training example in Figure 1.

moved, yielding the reusable linguistic resources in the grammar, as illustrated in Figure 2, as well as the maximum likelihood estimates needed to compute operation probabilities.

Our approach uses this induced grammar to treat generation as a search problem: given a desired semantic representation $M' \subseteq M$, use the grammar to incrementally construct an output utterance u that expresses M'. We treat generation as anytime search by accruing multiple goal states up until a specified timeout (for Doctor Perez: 200ms) and returning a list of alternative outputs ranked by their derivation probabilities.

3.3 Optimizing the Search Strategy

The search space created by a grammar induced in this way is too large to be searched exhaustively in most applications. The solution we have developed is a beam search strategy that uses weighted features to rank alternative grammatical expansions at each step. In particular, the beam size and structure is optimized so that, with high probability, the beam can be searched exhaustively before the timeout.[1] The second step of automated processing, then, is a training problem of finding weighted features such that for every training problem, nodes that lead to good generation output are ranked highly enough by those features to make it into the beam.

We use domain-independent rules to automatically define a set of features that could be heuristically useful for a given induced grammar. These include features for various syntactic structures and operations, numbers of undesired and desired meanings of different types added by an expansion, derivation probabilities, etc. (For Doctor Perez, this yields about 600 features.) Our training algorithm is based on the search optimization algorithm of (Daumé and Marcu, 2005), which updates feature weights when mistakes are made during search on training examples. For the weight update step, we use the boosting approach of (Collins and Koo, 2005), which performs feature selection and identifies weight values that improve the ranking of alternative derivation steps when mistakes are made during search. We discuss the resulting success rate and quality in the next section.

4 Cost/Benefit Analysis

The motivation that underlies our technical approach is to reduce the development costs and linguistic expertise needed to develop a grammar-based generation component for an existing system. In this section, we assess the progress we have made by ana-

[1]For Doctor Perez, we use a wider beam for initial trees, since the Doctor's semantic representation is particularly impoverished at the level of main verbs. At search depths > 1, we use beam size 1 (i.e. greedy search).

lyzing the use of our approach for Doctor Perez.

Method. We began with a sample of 220 instances of frames that Doctor Perez's dialogue manager had requested of the generation component in previous dialogues with users. Each frame was associated with a hand-authored target output utterance. We then constructed two alternative training examples, in the format specified in Section 3.1, for each frame. One example had uncorrected output of the Charniak parser for the syntax, and another had hand-corrected parser output (see Figure 1). The connections between surface expressions and frame key-value pairs were identical in both uncorrected and corrected training sets.

We then built two generators using the two sets of training examples. We used 90% of the data for training and held out 10% for testing. The generators sometimes failed to find a successful utterance within the 200ms timeout. For example, the success rate of the version of our generator trained on uncorrected syntax was 96.0% for training examples and 81.8% for test examples.

Quality of generated output. To assess output quality, 5 system developers rated each of 494 utterances, in relation to the specific frame for which it was produced, on a single 1 ("very bad") to 5 ("very good") scale. The 494 utterances included all of the hand-authored (suggested) utterances in the training examples. They also included all the top-ranked utterances that were successfully generated by the two generators. We asked our judges to make an overall assessment of output quality, incorporating both accuracy and fluency, for the Doctor Perez character. Judges were blind to the conditions under which utterances were produced. We discuss additional details of this rating task in (DeVault et al., 2008).

The judges achieved a reliability of $\alpha = 0.708$ (Krippendorff, 1980); this value shows that agreement is well above chance, and allows for tentative conclusions. We ran a small number of planned comparisons on these ratings. Surprisingly, we found no significant difference between generated output trained on corrected and uncorrected syntax ($t(29) = 0.056, p > 0.9$ on test items, $t(498) = -1.1, p > 0.2$ on all items).[2] However, we did find that hand-authored utterances (mean rating 4.4) are significantly better ($t(388) = 5.9, p < 0.001$) than generated utterances (mean rating 3.8 for uncorrected syntax). These ratings are depicted in Figure 3. While the figure suggests a slight reduction in quality for generated output for test frames vs. training frames, we did not find a significant difference between the two ($t(19) = 1.4228, p > 0.15$).

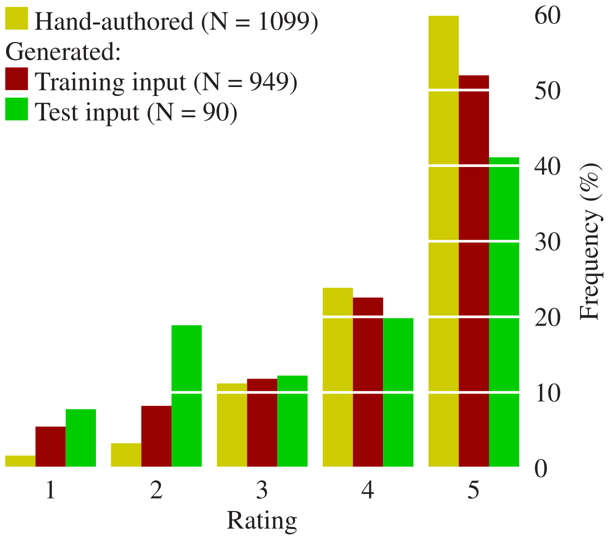


Figure 3: Observed rating frequencies for hand-authored vs. generated utterances (uncorrected syntax).

Variety of generated output. In general our anytime algorithm delivers a ranked list of alternative outputs. While in this initial study our judges rated only the highest ranked output generated for each frame, we observed that many of the lower ranked outputs are of relatively high quality. For example, Figure 4 shows a variety of alternative outputs that were generated for two of Doctor Perez's training examples. Many of these outputs are not present as hand-authored utterances (for any frame); this illustrates the potential of our approach to provide a variety of alternative outputs or paraphrases, which in some applications may be useful even for meanings for which an example utterance is hand-authored. Figure 5 shows the overall distribution in the number of outputs returned for Doctor Perez.

Development costs. The development costs included implementation of the approach and specification of Doctor Perez's training set. Implementa-

[2]The distribution of ratings across utterances is not normal; to validate our results we accompanied each t-test by a non-parametric Wilcoxon rank sum test, and significance always fell in the same general range.

Rank	Time (ms)	Novel?	
1	16	no	*the clinic is up to standard captain*
2	94	no	*the clinic is acceptable captain* (hand-authored for this input)
3	78	yes	*the clinic should be in acceptable condition captain*
4	16	yes	*the clinic downtown is currently acceptable captain*
5	78	yes	*the clinic should agree in an acceptable condition captain*
1	94	no	*there are no medical supplies downtown captain*
2	172	no	*we don't have medical supplies downtown captain*
3	125	yes	*well captain i do not think downtown there are medical supplies*
4	16	yes	*i do not think there are medical supplies downtown captain*

Figure 4: All the utterances generated (uncorrected syntax) for two examples. Rank is determined by derivation probability. Outputs marked as novel are different from any suggested output for any training example.

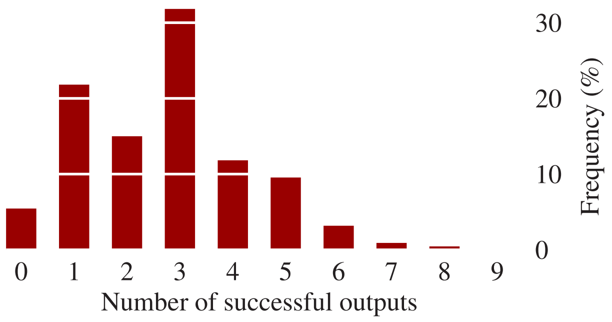


Figure 5: Variety of outputs for each input.

tion required an effort of approximately six person months. The developer who carried out the implementation initially had no familiarity with the Doctor Perez domain, so part of this time was spent understanding Doctor Perez and his available semantic representations. The bulk of the development time was spent implementing the grammar induction and training processes. Grammar induction included implementing the probabilistic grammar model and writing about 40 rules that are used to extract grammatical entries from the training examples. Of these 40 rules, only 3 are specific to Doctor Perez.[3] The remainder are broadly applicable to syntactic analyses in Penn Treebank format, and thus we expect they would transfer to applications of our approach in other domains. Similarly, the training algorithms are entirely domain neutral and could be expected to transfer well to additional domains.

Specification of Doctor Perez's training data took about 6 hours, or about 1.6 minutes per training example. This time included hand correction of syntactic analyses generated by the Charniak parser and definition of semantic links between surface expressions and frame key-value pairs. Since we found that hand-correcting syntax does not improve output quality, this 1.6 minutes/example figure overestimates the authoring time required by our approach. The remaining work lies in defining the semantic links. For Doctor Perez, approximately half of the semantic links were automatically assigned with simple ad hoc scripts.[4] The semantic linking process might be further sped up through a streamlined authoring interface offering additional automation, or even using a machine learning approach to suggest appropriate links.

Linguistic expertise required. Since we found that hand-correcting syntax does not improve output quality, a developer who wishes to exploit our approach may use the Charniak parser to supply the syntactic model for the domain. Thus, while one developer with linguistic expertise is required to implement the approach, anybody on the application team can contribute by hand authoring additional utterances and defining semantic links. The benefit of this authoring effort is the ability to generate high quality output for many novel semantic inputs.

Cost/benefit. The grammar induced from the 198 training examples (with uncorrected syntax) contains 426 lexical entries of the type depicted in Figure 2. These 426 lexical entries were produced automatically from about 6 hours worth of authoring ef-

[3]These 3 rules compensate for frequent errors in Charniak parser output for the words *captain*, *elder*, and *imam*, which are often used to signal the addressee of Doctor Perez's utterances.

[4]Time to compose these scripts is included in the 1.6 minutes/example.

fort together with domain-neutral algorithms. This translates to a rate of grammar expansion of less than 1 minute per lexical entry, on average, for this small application-specific grammar. This constitutes a dramatic improvement over our previous experience hand-crafting grammars. It would be challenging for an expert to specify a lexical entry such as those in Figure 2 in under one minute (and probably impossible for someone lacking detailed linguistic knowledge). In our experience, however, the bulk of development lies in additional time spent considering and investigating possible interactions between lexical entries in generation. Our technique helps with both problems: the grammar induction streamlines the specification of lexical entries, and the training removes the need for a developer to manually trace through the various complex interactions between lexical entries during generation.

5 Limitations

Currently, we do not support semantic links from non-contiguous expressions, which means a desired output like "we rely heavily on medical supplies" would be difficult to annotate if *rely...on* corresponds to a single semantic representation. This is not an intrinsic limitation to our general approach, but rather a simplification in our initial implementation.

As discussed in Section 3.2, our grammar induction process adds syntactic features related to verb inflection, finiteness, and grammatical role to the inferred lexical entries. Such features improve the fluency and accuracy of output derived with the grammar. While we believe such features can always be assigned using domain-independent rules, developing these rules requires linguistic expertise, and it is likely that additional rules and features (not yet implemented) would improve coverage of linguistic phenomena such as control verbs, various kinds of coordination, and relative clauses, inter alia.

A more entrenched limitation of our approach is its assumption that the generator does not need context as a separate input. This means, for example, that our approach cannot generate referring expressions (by selecting disambiguating semantic properties); rather, all semantic properties must be pre-selected and included in the generation request. Generation of anaphoric expressions is also limited, since contextual ambiguities are not considered.

6 Related Work

To our knowledge, this is the first implemented generation technique that does all three of the following: directly interfaces to existing application semantic representations, infers a phrase structure grammar from examples, and does not require hand-authored syntax as input. (Varges and Mellish, 2001) also aims to reduce the authoring burden of domain-specific generation; however, they seem to use a special purpose semantic annotation rather than pre-existing application semantics, and their task is defined in terms of the Penn Treebank, so hand-authored syntax is used as input. (Wong and Mooney, 2007) also interfaces to existing application semantics, and does not require hand-authored syntax as input. Their technique infers a synchronous grammar in which the hierarchical linguistic analysis is isomorphic to the hierarchy in the application semantics, and differs from phrase structure. It would be interesting to compare their output quality with ours; their automated alignment of words to semantics might also provide a way to further reduce the authoring burden of our approach.

7 Conclusion and Future Work

We have presented a new example-based approach to specifying text generation for an existing application. We have used a cost/benefit analysis to argue that our approach offers productive coverage and high-quality output with less linguistic expertise and lower development costs than building a hand-crafted grammar. In future work, we will evaluate our approach in additional application settings, and study the performance of our approach as the size and scope of the training set grows.

Acknowledgments

Thanks to our anonymous reviewers, Arno Hartholt, Susan Robinson, Thomas Russ, Chung-chieh Shan, and Matthew Stone. This work was sponsored by the U.S. Army Research, Development, and Engineering Command (RDECOM), and the content does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

References

S. Busemann and H. Horacek. 1998. A flexible shallow approach to text generation. In *Proceedings of INLG*, pages 238–247.

Aoife Cahill and Josef van Genabith. 2006. Robust PCFG-based generation using automatically acquired LFG approximations. In *ACL*, pages 1033–1040.

C. B. Callaway. 2003. Evaluating coverage for large symbolic NLG grammars. *Proceedings of IJCAI*.

E. Charniak. 2001. Immediate-head parsing for language models. In *ACL*, pages 124–131, Morristown, NJ, USA. Association for Computational Linguistics.

E. Charniak. 2005. ftp://ftp.cs.brown.edu/pub/nlparser/parser05Aug16.tar.gz.

D. Chiang. 2000. Statistical parsing with an automatically-extracted tree adjoining grammar. In *ACL '00: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 456–463, Morristown, NJ, USA. Association for Computational Linguistics.

D. Chiang. 2003. Statistical parsing with an automatically extracted tree adjoining grammar. In R. Bod, R. Scha, and K. Sima'an, editors, *Data Oriented Parsing*, pages 299–316. CSLI Publications, Stanford.

M. Collins and T. Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70.

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. dissertation, University of Pennsylvania.

H. Daumé and D. Marcu. 2005. Learning as search optimization: approximate large margin methods for structured prediction. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 169–176, New York, NY, USA. ACM.

David DeVault, David Traum, and Ron Artstein. 2008. Making grammar-based generation easier to deploy in dialogue systems. In *Ninth SIGdial Workshop on Discourse and Dialogue (SIGdial)*.

M. Elhadad. 1991. FUF: the universal unifier user manual version 5.0. Technical Report CUCS-038-91.

K. Krippendorff, 1980. *Content Analysis: An Introduction to Its Methodology*, chapter 12, pages 129–154. Sage, Beverly Hills, CA.

I. Langkilde and K. Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *COLING-ACL*, pages 704–710.

I. Langkilde-Geary. 2002. An empirical verification of coverage and correctness for a general-purpose sentence generator.

D. M. Magerman. 1995. Statistical decision-tree models for parsing. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 276–283, Morristown, NJ, USA. Association for Computational Linguistics.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1994. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

A. Stent, R. Prasad, and M. Walker. 2004. Trainable sentence planning for complex information presentation in spoken dialog systems. In *ACL*.

Matthew Stone, Doug DeCarlo, Insuk Oh, Christian Rodriguez, Adrian Stere, Alyssa Lees, and Chris Bregler. 2004. Speaking with hands: creating animated conversational characters from recordings of human performance. *ACM Trans. Graph.*, 23(3):506–513.

M. Stone. 2002. Lexicalized grammar 101. In *ACL Workshop on Tools and Methodologies for Teaching Natural Language Processing*.

Matthew Stone. 2003. Specifying generation of referring expressions by example. In *AAAI Spring Symposium on Natural Language Generation in Spoken and Written Dialogue*, pages 133–140.

W. Swartout, J. Gratch, R. W. Hill, E. Hovy, S. Marsella, J. Rickel, and D. Traum. 2006. Toward virtual humans. *AI Mag.*, 27(2):96–108.

D. R. Traum, W. Swartout, J. Gratch, and S. Marsella. 2008. A virtual human dialogue model for non-team interaction. In L. Dybkjaer and W. Minker, editors, *Recent Trends in Discourse and Dialogue*. Springer.

D. Traum. 2003. Semantics and pragmatics of questions and answers for dialogue agents. In *proceedings of the International Workshop on Computational Semantics*, pages 380–394, January.

Sebastian Varges and Chris Mellish. 2001. Instance-based natural language generation. In *NAACL*, pages 1–8.

M. Walker, O. Rambow, and M. Rogati. 2001. Spot: A trainable sentence planner. In *Proceedings of the North American Meeting of the Association for Computational Linguistics*.

M. White, R. Rajkumar, and S. Martin. 2007. Towards broad coverage surface realization with CCG. In *Proc. of the Workshop on Using Corpora for NLG: Language Generation and Machine Translation (UC-NLG+MT)*.

Yuk Wah Wong and Raymond Mooney. 2007. Generation by inverting a semantic parser that uses statistical machine translation. In *Proceedings of NAACL-HLT*, pages 172–179.

H. Zhong and A. Stent. 2005. Building surface realizers automatically from corpora using general-purpose tools. In *Proc. Corpus Linguistics '05 Workshop on Using Corpora for Natural Language Generation*.