Stochastic Language Generation in Dialogue Using Factored Language Models

François Mairesse*
University of Cambridge

Steve Young**
University of Cambridge

Most previous work on trainable language generation has focused on two paradigms: (a) using a statistical model to rank a set of pre-generated utterances, or (b) using statistics to determine the generation decisions of an existing generator. Both approaches rely on the existence of a handcrafted generation component, which is likely to limit their scalability to new domains. The first contribution of this article is to present BAGEL, *a fully data-driven generation method that treats the language generation task as a search for the most likely sequence of semantic concepts and realization phrases, according to Factored Language Models (FLMs). As domain utterances are not readily available for most natural language generation tasks, a large creative effort is required to produce the data necessary to represent human linguistic variation for nontrivial domains. This article is based on the assumption that learning to produce paraphrases can be facilitated by collecting data from a large sample of untrained annotators using crowdsourcing—rather than a few domain experts—by relying on a coarse meaning representation. A second contribution of this article is to use crowdsourced data to show how dialogue naturalness can be improved by learning to vary the output utterances generated for a given semantic input. Two data-driven methods for generating paraphrases in dialogue are presented: (a) by sampling from the n-best list of realizations produced by* BAGEL*'s FLM reranker; and (b) by learning a structured perceptron predicting whether candidate realizations are valid paraphrases. We train* BAGEL *on a set of 1,956 utterances produced by 137 annotators, which covers 10 types of dialogue acts and 128 semantic concepts in a tourist information system for Cambridge. An automated evaluation shows that* BAGEL *outperforms utterance class LM baselines on this domain. A human evaluation of 600 resynthesized dialogue extracts shows that* BAGEL*'s FLM output produces utterances comparable to a handcrafted baseline, whereas the perceptron classifier performs worse. Interestingly, human judges find the system sampling from the n-best list to be more natural than a system always returning the first-best utterance. The judges are also more willing to interact with the n-best system in the future. These results suggest that capturing the large variation found in human language using data-driven methods is beneficial for dialogue interaction.*

* The author's present address is Amazon.com, 101 Main Street, Suite 900, Cambridge, MA 02142, USA. E-mail: francois.mairesse@gmail.com. This research was done at the University of Cambridge.
** Cambridge University Engineering Department, Trumpington Street, Cambridge, CB2 1PZ, UK. E-mail: sjy@eng.cam.ac.uk.

Submission received: 12 June 2011; revised version received: 12 November 2013; accepted for publication: 21 December 2013.

doi:10.1162/COLI_a_00199

1. Introduction

The field of natural language generation (NLG) was one of the last areas of computational linguistics to embrace statistical methods, perhaps because of the difficulty of collecting semantically annotated corpora. Over the past decade, statistical NLG has followed two lines of research. The first, pioneered by Langkilde and Knight (1998), introduces statistics in the generation process by training a model that reranks candidate outputs of a handcrafted generator. Their HALOGEN system uses an *n*-gram language model trained on news articles. HALOGEN is thus domain-independent, and it was successfully ported to a specific dialogue system domain (Chambers and Allen 2004). However, its performance depends largely on the granularity of the underlying meaning representation, which typically includes syntactic and lexical information. A major issue with data-driven NLG systems is that collecting fine-grained semantic annotations requires a large amount of time and expertise. For most domains, handcrafting templates remains a more cost-effective solution.

More recent work has investigated other types of reranking models, such as hierarchical syntactic language models (Bangalore and Rambow 2000), discriminative models trained to replicate user ratings of utterance quality (Walker, Rambow, and Rogati 2002), or language models trained on speaker-specific corpora to model linguistic alignment (Isard, Brockmann, and Oberlander 2006). However, a major drawback of the utterance-level overgenerate and rank approach is its inherent computational cost. In contrast, this article proposes a method in which local overgeneration can be made tractable through beam pruning.

A second line of research has focused on introducing statistics at the generation-decision level by training models that find the set of generation parameters maximizing an objective function, for example, producing a target linguistic style (Paiva and Evans 2005; Mairesse and Walker 2008), generating the most likely context-free derivations given a corpus (Belz 2008), or maximizing the expected reward using reinforcement learning (Rieser and Lemon 2010). Although such methods do not suffer from the computational cost of an overgeneration phase, they still require a handcrafted generator to define the generation decision space within which statistics can be used to find an optimal solution. Recently, research has therefore focused on reducing the amount of handcrafting required by learning to infer generation rules from data (see Section 2).

This article presents BAGEL, an NLG system that can be fully trained from utterances aligned with coarse-grained semantic concepts. BAGEL aims to produce natural utterances within a large dialogue system domain while minimizing the overall development effort. Because repetitions are common in human–computer interactions—especially when facing misunderstandings—a secondary objective of this article is to improve dialogue naturalness by learning to generate *paraphrases* from data. Although domain experts can be used to annotate data, domain utterances are not readily available for most NLG tasks, hence a creative process is required for generating these utterances as well as matching semantics. The difficulty of this process is increased for systems aiming at producing a large amount of linguistic variation, because it requires enumerating a large set of paraphrases for each domain input. This article is based on the assumption that learning to produce paraphrases can be facilitated by collecting data from a large sample of annotators. However, this requires that the meaning representation should (a) be simple enough to be understood by *untrained* annotators, and (b) provide useful generalization properties for generating unseen inputs. Section 3 describes BAGEL's meaning representation, which satisfies both requirements. Section 4 then details how our meaning representation is mapped to a phrase sequence, using

cascaded Factored Language Models with back-off smoothing. Section 5 presents two methods for using BAGEL's probabilistic output for paraphrase generation in dialogue. Section 6 illustrates how semantically aligned training utterances for a large tourist information domain were collected using crowdsourcing. Section 7 then evaluates the trained models in a dialogue setting, by showing that (a) BAGEL performs comparably to a handcrafted rule-based generator; and (b) human judges prefer systems sampling from the *n*-best output over systems always selecting the top ranked utterance. Finally, Section 8 discusses the implication of these results as well as future work.

2. Related Work

Although statistics have been widely used to tune NLG systems, most previous work on trainable NLG has relied on a pre-existing handcrafted generator (Langkilde and Knight 1998; Walker, Rambow, and Rogati 2002). Only recently has research started to develop NLG models trained from scratch, without any handcrafting beyond the definition of the semantic annotations.

In order to reduce complexity, previous work has split the NLG task into two phases: (a) sentence planning and (b) surface realization. The sentence planning phase maps input semantic symbols to an intermediary tree-like or template structure representing the utterance; then the surface realization phase converts it into the final text. As developing a sentence planner capable of overgeneration typically requires a substantial amount of handcrafting (Walker, Rambow, and Rogati 2002; Stent, Prasad, and Walker 2004), Stent and Molina (2009) have proposed a method that learns sentence planning rules from a corpus of utterances labeled with Rhetorical Structure Theory (RST) discourse relations (Mann and Thompson 1988). Although additional handcrafting is needed to map the sentence plan to a valid syntactic form by aggregating the syntactic structures of the relations arguments, we believe RST offers a promising framework for improving the expressiveness of statistical generators. Section 8 discusses how BAGEL's expressiveness could be improved by including RST relations.

Language models (LMs) have previously been used for language generation in order to remove the need for a handcrafted overgeneration phase (Oh and Rudnicky 2002; Ratnaparkhi 2002). Oh and Rudnicky's (O&R) approach trains a set of word-based *n*-gram LMs on human–human dialogues, one for each **utterance class** in their corpus. An utterance class corresponds to the intent and zero or more slots in the input dialogue act. At generation time, the corresponding LM is used for overgenerating a set of candidate utterances, from which the final utterance is selected based on a set of reranking rules. Ratnaparkhi addresses some limitations of the overgeneration phase by comparing systems casting the NLG task as (a) a search over a word sequence based on an *n*-gram probabilistic model, and (b) as a search over syntactic dependency trees based on models predicting words given its syntactic parent and sibling nodes (Ratnaparkhi 2002). O&R's method represents the first line of research on NLG that limits the amount of handcrafting to a small set of post-processing rules in order to facilitate the development of a dialogue system's NLG component. Section 7.1 therefore compares BAGEL's performance with O&R's utterance class LM approach, and discusses differences between the two techniques.

Data-driven NLG research has also been inspired by research on semantic parsing and machine translation. The WASP^{-1} generator combines a language model with an inverted synchronous context-free grammar parsing model, effectively casting the generation task as a translation problem from a meaning representation to natural language (Wong and Mooney 2007). WASP^{-1} relies on GIZA++ to align utterances with

derivations of the meaning representation (Och and Ney 2003). Although early experiments showed that GIZA++ did not perform well on our data—possibly because of the coarse granularity of our semantic representation—future work should evaluate the generalization performance of synchronous context-free grammars in a dialogue system domain. Lu, Ng, and Lee (2009) show that Tree Conditional Random Fields (CRFs) outperform WASP^{-1} and their own inverted semantic parser, based on automated evaluation metrics, although their system remains to be evaluated by human judges (Lu, Ng, and Lee 2009). Similarly to the perceptron reranking approach presented here, Tree CRFs learn a log linear model, estimating the conditional probability of semantic tree/phrase alignments given an input semantic tree. Although this line of research is promising, the two data sets evaluated—GEOQUERY and ROBOCUP—contain a large number of utterances that only differ by the proper name used. For example, 17 out of the 880 instances of the GEOQUERY data set match the template *what is the capital of $STATE*. Such instances are therefore likely to occur simultaneously in the training and test partitions. In contrast, in our evaluation such templates are mapped to the same meaning representation, and we enforce the condition that the generated meaning representation was not seen during training.

Angeli, Liang, and Klein (2010) propose a simpler framework in which the generation task is cast as a sequence of generation decisions selecting either: (a) a database record to express (e.g., the *temperature*); (b) a set of fields for that record (e.g., the *minimum*, *maximum*); and (c) a template realizing those fields (e.g., *with a low around $MINIMUM*). They train a set of log-linear models predicting individual generation decisions given the previous ones, using domain-independent features capturing the lexical context as well as content selection. The templates are extracted from data aligned with the input records using expectation maximization. This approach offers the benefit of allowing predictions to be made given generation decisions that are arbitrarily far in the past. However, long-range feature dependencies make a Viterbi search intractable, hence the authors use a greedy search, which produces state-of-the-art results on the ROBOCUP data set and two weather domains. More recently, Kondadadi, Howald, and Schilder (2013) also decouple the NLG task as a template extraction and ranking problem, and show that an SVM reranker can produce outputs comparable to human-authored texts for weather reports and short biographies.[1]

Konstas and Lapata (2012) jointly model content selection and surface realization by training a forest of PCFGs expressing the relation between records, fields, and words. A Viterbi search is used to find the optimal derivations at generation time; however, the PCFG weights are rescored using an averaged structured perceptron using both content selection and lexical features. The authors show that their approach outperforms Angeli, Liang, & Klein's (2010) method on the air transport query domain (ATIS data set). This article evaluates the same averaged structured perceptron algorithm within the BAGEL framework (see Sections 5.2 and 7.2).

Most other work on data-driven NLG has focused on learning to map syntax to text. The surface realization task is an attractive research topic as it is not tied to a specific application domain. Factored language models have been used for surface realization within the OpenCCG framework (White, Rajkumar, and Martin 2007; Espinosa, White, and Mehay 2008). More generally, chart generators for different grammatical formalisms have been trained from syntactic treebanks (Nakanishi, Miyao, & Tsujii 2005; Cahill and

1 These papers were published after the main BAGEL development and thus no detailed comparisons are offered in this article.

van Genabith 2006; White, Rajkumar, and Martin 2007), as well as from semantically annotated treebanks (Varges and Mellish 2001). Because manual syntactic annotation is costly and syntactic parsers do not necessarily perform well at labeling spoken language utterances, the present work focuses on the generation of surface forms directly from semantic concepts. Future work should investigate whether explicit syntactic modeling improves performance (e.g., by conditioning the realization FLMs on part-of-speech information).

Previous studies have shown that paraphrasing improves performance in automated tutoring dialogues (Pon-Barry et al. 2006), and suggested that users prefer dialogue systems in which repetitions are signaled (e.g., *as I said before*), even though that preference was not significant (Foster and White 2005). However, we do not know of any research applying statistical paraphrasing techniques to dialogue. Most research on paraphrasing has focused on unsupervised techniques for extracting paraphrases from a corpus of written text. Proposed techniques learn to identify phrase templates, which tend to have the same arguments in a monolingual corpus (Lin and Pantel 2001), or to detect variations between translations of the same text (Barzilay and McKeown 2001; Bannard and Callison-Burch 2005). Although these methods could be used to enrich an existing generator, they do not model semantics; hence they cannot be applied directly to NLG. Statistical reranking models have been used for over a decade for language generation (Langkilde and Knight 1998); however, we do not know of any evaluation of their paraphrasing power. Whereas linguistic variation is typically ignored in NLG systems, a recent line of research has started investigating how to control a generator to convey a specific style—for example, to generate language with a target linguistic genre (Paiva and Evans 2005), to convey a specific personality trait (Mairesse and Walker 2008, 2011), or to align with their conversational partner (Isard, Brockmann, and Oberlander 2006). These systems use statistics for controlling the style of their output; however, they require an existing handcrafted generator, and they were not evaluated within a dialogue context. We believe that the techniques presented here can also be used for stylistic control by including stylistic elements in our stack-based semantic representation; however, we leave this to future work.

Another line of work has used NLG paraphrase mechanisms to show that jointly optimizing NLG and speech synthesis can improve human perceptions of voice quality. This was achieved by finding the candidate paraphrase yielding the lowest speech unit concatenation cost using weighted finite state transducers (Bulyko and Ostendorf 2002) or by using a discriminative reranker trained to predict human judgments of synthesis quality (Nakatsu and White 2006). Similarly, Stone et al. (2004) propose a method using dynamic programming for simultaneously optimizing NLG, speech synthesis, and gesture in animated characters. Although all three approaches learn the paraphrase selection step from data, they rely on handcrafted NLG for producing candidates. Hence future work should investigate whether voice quality could also be improved by composing the n-best paraphrases generated by BAGEL with a prosodic reranker.

3. Phrase-Based Generation from Semantic Stacks

BAGEL uses a *stack-based* semantic representation to constrain the sequence of semantic concepts to be searched. This representation can be seen as a linearized semantic tree similar to the one previously used for natural language understanding in the Hidden Vector State model (He and Young 2005). A stack representation provides useful generalization properties, and it allows for efficient sequential decoding using dynamic programming. In the context of dialogue systems, Figures 1 and 2 illustrate how the

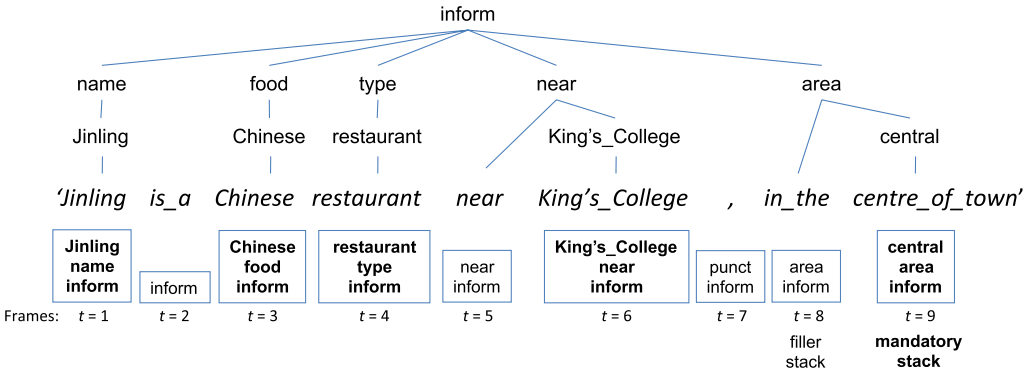


Figure 1
Example utterance for the `inform` dialogue act type, with aligned semantic tree and corresponding stack sequence in boxes. Mandatory stacks are in **bold**.

input dialogue act (i.e., a semantic tree) is mapped to a set of stacks of semantic concepts (represented as boxes) and aligned with a phrase sequence, resulting in one stack/phrase pair per time frame. The root concept of the semantic tree (i.e., the bottom concept in each stack) expresses the overall communicative goal of the utterance and is referred to as a **dialogue act type**. For example, the `inform` dialogue act type in Figure 1 indicates that the utterance provides information about an entity matching the user's constraints; the dialogue act type `informall` in Figure 2 indicates that all the entities matching some of the user's constraints also satisfy other constraints. In contrast, the `reject` dialogue act type indicates that the system cannot find an entity matching the specified constraints. See Table 4 in Section 6 for more example dialogue act types. Non-root semantic concepts include attributes of that entity under consideration (e.g., `name`, `food`, and `area` at frame 1, 3, and 9 in Figure 1), values for those attributes (e.g., respectively, `name(Jinling)`, `food(Chinese)`, and `area(centre)` in Figure 1), as well as special symbols for logical quantifiers (e.g., `all` in Figure 2), negations (`not`), or

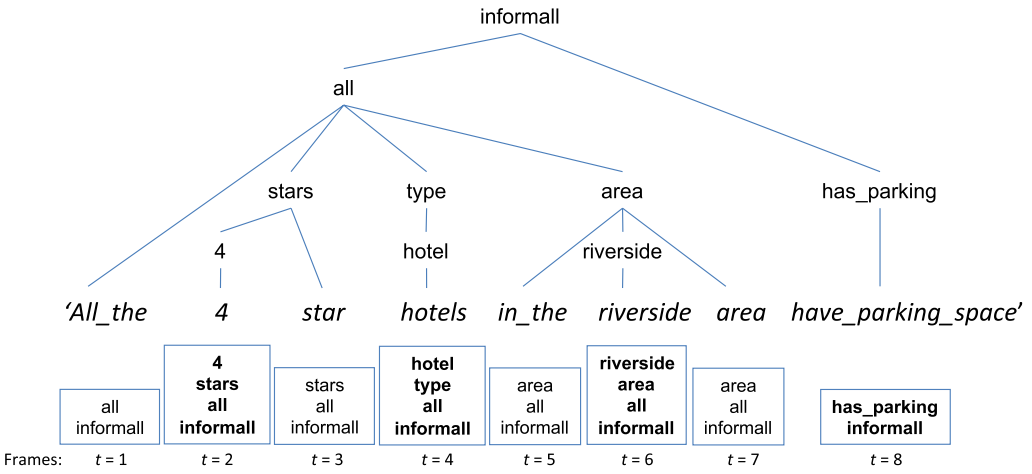


Figure 2
Example utterance for the `informall` dialogue act type, with aligned semantic tree and corresponding stack sequence in boxes. Mandatory stacks are in **bold**.

specifying that an attribute is irrelevant (`dontcare`). Punctuation symbols are modeled using the semantic concept `punct`, as in frame 7 in Figure 1.

The generator's goal is thus to find the most likely realization given an *unordered* collection of *mandatory* semantic stacks $\mathcal{S}_m$ derived from the input dialogue act. Mandatory stacks are represented in bold in Figure 1, such as `inform(area(centre))` in frame 9. While mandatory stacks must all be conveyed in the output realization, $\mathcal{S}_m$ does not contain the optional *filler* stacks $\mathcal{S}_i$ that can refer to (a) general attributes of the object under discussion (e.g., `inform(area)` in frame 8); (b) concepts that are not in the input at all, which are associated with the singleton stack `inform` (e.g., phrases specific to a dialogue act type such as *'is a'* in Figure 1, or clause aggregation operations such as *'and'*); or (c) punctuation symbols (e.g., `inform(punct)` in frame 7).

The general philosophy behind our semantic formalism is to match the practical requirements of an information presentation dialogue system; that is, a dialogue manager typically returns a tree-like structure of coarse-grained semantic concepts describing (a) the overall dialogue act type, (b) the constraints over entities stored in a domain-specific database, as well as (c) logical modifiers expressing relations between sets of domain entities, depending on the dialogue act type. A major advantage of our formalism compared with more fine-grained formalisms (e.g., lambda calculus) is that it can be easily understood by human annotators. We believe that this is a crucial point for collecting the range of utterances required for learning to generate natural paraphrases in large domains (see Section 6). Furthermore, Section 8 discusses how its expressiveness could be extended by including additional discourse structures.

BAGEL's granularity is defined by the semantic annotation in the training data, rather than external linguistic knowledge about what constitutes a unit of meaning; namely, contiguous words belonging to the same semantic stack are modeled as an atomic observation unit or **phrase**.[2] In contrast with word-level language models, a major advantage of phrase-based generation models is that they can model long-range dependencies and domain-specific idiomatic phrases with fewer parameters.

4. FLM-Based Statistical NLG

In order to find the optimal stack and realization phrase sequences given an input dialogue act, we cast the generation task as a search over Factored Language Models (FLMs), which were introduced by Bilmes and Kirchhoff (2003). FLMs extend traditional language models by allowing predicted variables to be conditioned on different utterance contexts, depending on whether they were sufficiently observed in the training data. This approach is equivalent to a dynamic Bayesian network in which the probability of child nodes are estimated by interpolating over different parent nodes. Dynamic Bayesian networks have been used successfully for speech recognition, natural language understanding, dialogue management, and text-to-speech synthesis (Rabiner 1989; Tokuda et al. 2000; He and Young 2005; Lefèvre 2006; Thomson and Young 2010). Such models provide a principled framework for predicting elements in a large structured space, such as required for non-trivial NLG tasks. Additionally, their probabilistic nature makes them suitable for modeling linguistic variation—that is, there can be multiple valid paraphrases for a given input.

2 The term *phrase* is thus defined here as any sequence of one or more words.

4.1 NLG as a Viterbi-Search Pipeline

BAGEL models the generation task as finding the most likely sequence of realization phrases $\mathbf{R}^* = (r_1 ... r_L)$ given an input dialogue act. Each dialogue act is represented as a set of mandatory semantic stacks $\mathcal{S}_m$ (unordered), with $|\mathcal{S}_m| \leq L$. BAGEL must thus derive the optimal sequence of semantic stacks $\mathbf{S}^*$ that will appear in the utterance given $\mathcal{S}_m$, that is, by inserting filler stacks if needed and by performing content ordering. Let us define the set of mandatory stack orderings as $Order(\mathcal{S}_m)$. Any number of filler stacks can be inserted between two consecutive mandatory stacks, as long as all their concepts are included in either the previous or following mandatory stack, and as long as each stack transition leads to a different stack (see example in figures 1 and 2). For each mandatory stack sequence $\mathbf{S_m}$ in $Order(\mathcal{S}_m)$, let us define the set of all possible stack sequences matching the filler insertion constraints as $Fill(\mathbf{S_m})$.

Ideally, we would like to learn a model that estimates the probability of a realization given a dialogue act $P(\mathbf{R}|\mathcal{S}_m)$ from a training set of realization phrases aligned with semantic stack sequences. During the generation process, the realization probability can be computed by marginalizing over all possible semantic stack sequences satisfying the dialogue act constraints:

$$\begin{aligned} P(\mathbf{R}|\mathcal{S}_m) &= \sum_{\mathbf{S_m} \in Order(\mathcal{S}_m)} \sum_{\mathbf{S} \in Fill(\mathbf{S_m})} P(\mathbf{R}, \mathbf{S}, \mathbf{S_m}|\mathcal{S}_m) \\ &= \sum_{\mathbf{S_m} \in Order(\mathcal{S}_m)} \sum_{\mathbf{S} \in Fill(\mathbf{S_m})} P(\mathbf{R}|\mathbf{S}, \mathbf{S_m}, \mathcal{S}_m) P(\mathbf{S}|\mathbf{S_m}, \mathcal{S}_m) P(\mathbf{S_m}|\mathcal{S}_m) \\ &= \sum_{\mathbf{S_m} \in Order(\mathcal{S}_m)} P(\mathbf{S_m}|\mathcal{S}_m) \sum_{\mathbf{S} \in Fill(\mathbf{S_m})} P(\mathbf{R}|\mathbf{S}) P(\mathbf{S}|\mathbf{S_m}) \end{aligned} \quad (1)$$

Inference over such a model would require the decoding algorithm to consider all possible underlying stack sequences together with all possible realizations, which is intractable for non-trivial domains. Because a key requirement of this work was to develop data-driven techniques that can be used to generate utterances in *real-time*, the generation task is approximated by splitting it into three sequential decoding steps, illustrated in Figure 3:

1. The ordering of mandatory stacks $\mathbf{S_m}$ is predicted independently from the filler stacks and the realization phrases. This model can be seen as a high-level content ordering model. For example, it learns whether or not the information about the venue's area should follow the information

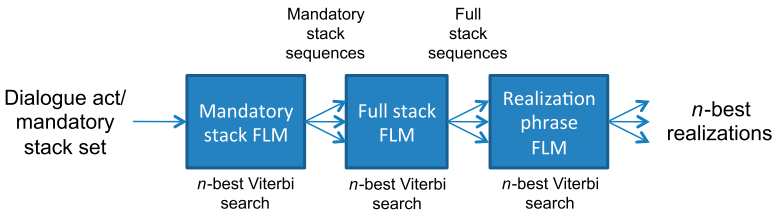


Figure 3
Architecture of an FLM-based statistical generator with three decoding stages.

about nearby venues. In order to limit the impact of this approximation, the top n sequences are selected as candidate inputs to the following step (argmax_N), rather than only the first-best. Hence the first generation step requires computing:

$$\mathbf{S}^*_\mathbf{m} = \underset{\mathbf{S_m} \in Order(\mathcal{S}_m)}{\text{argmax}_\text{N}} \; P(\mathbf{S_m}|\mathcal{S}_m) \tag{2}$$

2. The resulting n-best mandatory stack sequences $\mathbf{S}^*_\mathbf{m}$ are used to constrain the search for the full stack sequence $\mathbf{S}$ (i.e., by inserting filler stacks between consecutive mandatory stacks). For example, given that the information about the area follows the information about nearby venues, the model might predict that the actual area needs to be introduced by an area-specific expression (see filler stack at time $t = 8$ in Figure 1). Hence for the second generation step we compute:

$$\mathbf{S}^* = \underset{\mathbf{S} \in Fill(\mathbf{S}^*_\mathbf{m})}{\text{argmax}_\text{N}} P(\mathbf{S}|\mathbf{S}^*_\mathbf{m}) \tag{3}$$

3. The resulting n-best full stack sequences $\mathbf{S}^*$ are used to condition the search for the realization phrase sequence $\mathbf{R}$. For example, the realization phrase model is likely to predict that *in the* and *centre of town* should be associated with the two semantic stacks characterizing the area (see phrases at $t = 8$ and $t = 9$ in Figure 1). Hence the third generation step requires computing:

$$\mathbf{R}^* = \underset{\mathbf{R}=(r_1 \ldots r_L)}{\text{argmax}_\text{N}} P(\mathbf{R}|\mathbf{S}^*) \tag{4}$$

Each decoding step can be computed using dynamic programming; however, the decoding efficiency depends highly on the locality of context features. In the basic decoder, we factorize our models by conditioning the realization phrase at time t on the previous phrase r_{t-1}, and the previous, current, and following semantic stacks. The semantic stack s_t at time t is assumed to depend only on the previous two stacks:

$$P(\mathbf{S_m}|\mathcal{S}_m) = \begin{cases} \prod_{t=1}^{T} P(s_t|s_{t-1}, s_{t-2}) \\ \quad \text{if } \mathbf{S_m} \in Order(\mathcal{S}_m) \\ 0 \quad \text{otherwise} \end{cases} \tag{5}$$

$$P(\mathbf{S}|\mathbf{S}^*_\mathbf{m}) = \begin{cases} \prod_{t=1}^{T} P(s_t|s_{t-1}, s_{t-2}) \\ \quad \text{if } \mathbf{S} \in Fill(\mathbf{S}^*_\mathbf{m}) \\ 0 \quad \text{otherwise} \end{cases} \tag{6}$$

$$P(\mathbf{R}|\mathbf{S}^*) = \prod_{t=1}^{T} P(r_t|r_{t-1}, s^*_{t-1}, s^*_t, s^*_{t+1}) \tag{7}$$

As dynamic Bayesian networks typically require sequential inputs, some processing is needed to learn to map a *set* of semantic stacks to a phrase sequence. This is achieved by keeping track of the mandatory stacks that were visited in the current sequence and pruning any sequence that has not included all mandatory input stacks on

reaching the final frame. Because the number of filler stacks is not known at decoding time, the network is unrolled for a fixed number of frames T defining the maximum number of phrases that can be generated (e.g., $T = 50$). The end of the stack sequence is then determined by a special `end` symbol, which can only be emitted within the T frames once all mandatory stacks have been visited. The probability of the resulting utterance is thus computed over all frames up to the `end` symbol, which determines the length L of $\mathbf{S}^*$ and $\mathbf{R}^*$. Whereas the decoding constraints enforce that $L > |\mathcal{S}_m|$, the search for $\mathbf{S}^*$ requires comparing sequences of different lengths. A consequence is that shorter sequences containing only mandatory stacks are likely to be favored. Future work should investigate length normalization strategies, but we find that the learned transition probabilities are skewed enough to favor stack sequences that include filler stacks.

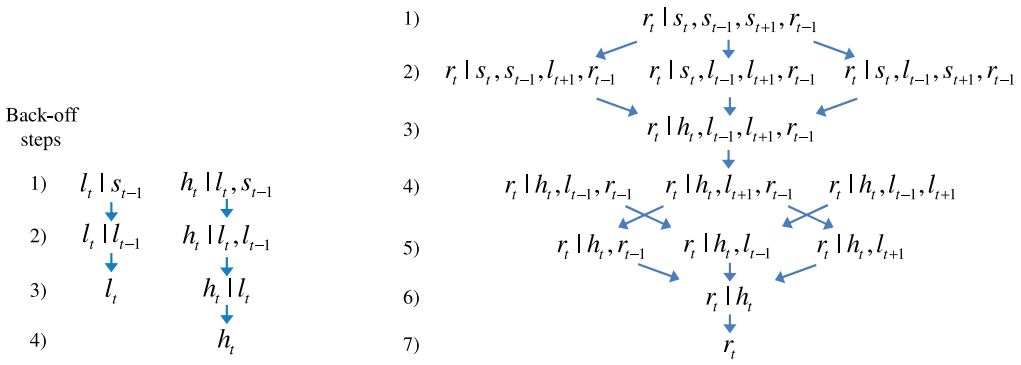
BAGEL solves Equations (2), (3), and (4) by doing three pipelined Viterbi searches to find the optimal sequence of output symbols (mandatory semantic stacks, filler stacks, and realization phrases) given the input (unordered mandatory stacks, ordered mandatory stacks, and the full stack sequence, respectively). Our initial generator thus consists of a pipeline of three FLMs, as illustrated in Figure 3. In terms of computational complexity, the number of stack sequences $Order(\mathcal{S}_m)$ to search over during the first decoding step increases exponentially with the number of input mandatory stacks. However, the proposed three-stage architecture allows for tractable decoding by (a) pruning low probability paths during each Viterbi search, and (b) pruning low probability sequences from the output n-best list of each component.

4.2 Generalization to Unseen Contexts

FLMs allow predicted symbols to be conditioned on any contextual feature. Furthermore, if a feature was not observed during training time, the FLM can *back off* to more general features according to a predefined back-off strategy. This section shows how the generation process can be made more robust to unseen dialogue acts by factoring the semantic stack and realization phrase variables.

4.2.1 Partial Stack Modeling. A robust language generator should be able to infer that some stack sequences are more likely than others even if they were only partially observed during training, based on co-occurrences on individual stack concepts. For example, such a generator should learn that `inform(type(restaurant))` is likely to follow `inform(pricerange(cheap))` based on the observation of `inform(pricerange(cheap))` followed by `inform(type(hotel))`. However, if `inform(type(restaurant))` has not been seen during training, it will be assigned a low probability regardless of its context. This can be alleviated by factorizing the stack variable into underspecified stack configurations—that is, model the probability of observing a stack s_t as the probability of observing the *tail* of the stack l_t as well as the *head* of the stack h_t given its tail. In other words, the probability of a stack occurrence given the previous stack is factorized as $P(s_t|s_{t-1}) = P(h_t, l_t|s_{t-1}) = P(l_t|s_{t-1})P(h_t|l_t, s_{t-1})$. As a result, the probability that `inform(pricerange(cheap))` is followed by `inform(type(restaurant))` will be high even if `inform(type(restaurant))` was not observed, as long as `inform(pricerange(cheap))` is frequently followed by the tail symbol `inform(type(SOMETHING))` in the training data.

Although the proposed factorization does not entirely resolve the data sparsity issue, it limits its impact to a single factor. In the example above, `inform(type(restaurant))` has not been seen during training; hence there is no data to estimate



(a) Semantic stack head h_t and tail l_t factor decomposition. The full stack probability is obtained by multiplying both factors using the chain rule.

(b) Realization phrase r_t given previous phrase r_{t-1} and semantic context s_{t-1}, s_t, s_{t+1}. At each parallel back-off step (2, 4, and 5), the probability of the most likely back-off path is used.

Figure 4
Back-off graphs for (a) the semantic stack FLMs and (b) the realization phrase FLM.

the probability that the head symbol `restaurant` governs `inform(type(SOMETHING))` in the second factor. A solution is to *back off* to the probability of `restaurant` occurring in a more general context (e.g., ignoring the underlying stack concepts). The back-off graphs of the two factors are illustrated in Figure 4(a), and an example sequence of back-off variables is shown in the right column of Table 1.

This example shows how a partial stack representation can improve semantic stack ordering, but it can also be used to assign non-zero probabilities to realization phrases observed in unseen semantic contexts by backing off to the head and the tail of the stack. This process is illustrated by the second and third back-off steps of the realization back-off graph in Figure 4(b). The neighboring semantic stacks s_{t-1} and s_{t+1} are first replaced by their stack tails l_{t-1} and l_{t+1}, respectively (step 2). If none of the three resulting contexts were observed during training, the current semantic stack s_t

Table 1
Example utterance annotation used to estimate the conditional probability distributions in figures 4 and 6 (r_t = realization phrase, s_t = semantic stack, h_t = stack head, l_t = stack tail).

r_t	s_t	h_t	l_t
<s>	START	START	START
The Rice Boat	inform(name(X))	X	inform(name)
is a	inform	inform	EMPTY
restaurant	inform(type(restaurant))	restaurant	inform(type)
in the	inform(area)	area	inform
riverside	inform(area(riverside))	riverside	inform(area)
area	inform(area)	area	inform
that	inform	inform	EMPTY
serves	inform(food)	food	inform
French	inform(food(French))	French	inform(food)
food	inform(food)	food	inform
</s>	END	END	END

is replaced by its stack head h_t (step 3). If this context was not observed either, the variables the farthest away are dropped in the following back-off steps. In extreme cases, the realization probability is approximated by the unigram count $P(r_t)$ in step 7. This mechanism provides BAGEL with the ability to generalize lexical realizations across contexts. For example, if `reject(area(centre))` was never observed at training time, $P(r_t = \text{centre of town} | s_t = \texttt{reject(area(centre))})$ can be estimated by backing off to $P(r_t = \text{centre of town} | h_t = \texttt{centre})$ in step 6. BAGEL can thus generate *there are no venues in the centre of town* if the phrase *centre of town* was associated with the concept `centre` in a different context, such as `inform(area(centre))`.

4.2.2 Partial Phrase Modeling. The robustness of FLM-based generation models can also be improved by allowing the realization model to back off to *partial phrase contexts*. For example, even if the phrase sequence *located in the* and *centre of town* has not been seen during training, it would be desirable for it to have a higher probability than *located in* followed by *centre of town*, which misses a determiner. This can be achieved by backing off to the last words of the preceding phrase (e.g., *in the* or *the*), which are more likely to precede *centre of town* in the data. Hence FLMs can learn to predict function words without allocating them an explicit time frame during decoding. In our experiments, we sequentially back off to the last two words and the last word of the preceding phrase.

The back-off graphs in Figure 4 illustrate the factorization and back-off strategies of BAGEL's decoding models, and Table 1 shows an instantiation of the back-off variables for an example utterance. The predictions of FLMs can be improved by *smoothing* their probability estimate over different contexts by interpolating between different back-off probability distributions (Bilmes and Kirchhoff 2003). In our experiments, the conditional probability distributions of the three models in Figure 3 are smoothed using Witten–Bell interpolated back-off smoothing, according to the back-off graphs in Figure 4. Generally, variables that are the farthest away in time are dropped first, and partial stack variables are dropped last, as they are observed the most. As the optimal back-off strategy can vary depending on the context, the realization model implements *parallel* back-off strategies (see steps 2, 4, and 5 in Figure 4(b))—that is, multiple back-off paths are explored at run-time, and the probability of each back-off node is computed as the maximum probability of all outgoing paths.

4.2.3 High Cardinality Concept Abstraction. Although we should expect a trainable generator to learn multiple lexical realizations for a given semantic concept, learning lexical realizations for high-cardinality database entries (e.g., proper names) would increase the number of model parameters prohibitively. We thus divide pre-terminal concepts in the semantic stacks into two types: (a) *enumerable* attributes whose values are associated with distinct semantic stacks in our model (e.g., `inform(pricerange(cheap))`), and (b) *non-enumerable* attributes whose values are replaced by a generic symbol before training in both the utterance and the semantic stack (e.g., `inform(name(X))`). These symbolic values are then replaced in the surface realization by the corresponding value in the input specification. A consequence is that our model can only learn synonymous lexical realizations for enumerable attributes.

4.3 Scaling to Large Domains Using Large-Context Reranking FLMs

A major inconvenience of the proposed approach is that the performance of the three Viterbi decoding steps is highly dependent on the size of the context of the predicted variable. For example, a trigram phrase model with a vocabulary of size V requires

searching over V symbols times V^2 paths leading to that symbol at every time step. Generating utterances for real-time interaction in a realistic domain typically limits context features to a single neighboring time frame (i.e., bigram) for both the semantic stack and realization models, which results in poor modeling accuracy. In order to model longer contexts while maintaining acceptable decoding performance, we use a *cascaded reranking* approach in which the n-best output of each Viterbi search is reranked by an FLM. The complexity of the reranking steps grows linearly with n and does not depend on V; hence its impact on performance is minimal compared with the decoding steps. Figure 5 illustrates the resulting pipeline of FLM models.

Early experimentation has led us to use the back-off strategies illustrated in Figure 6 for our reranking models, as they offer a rich context while maintaining acceptable real-time performance. The semantic reranking models are dependent on three preceding time frames, and the realization reranking model is dependent on the two previous and two following phrases. Reranking back-off strategies can be more complex than the strategies used during search, as they are only called over a small number of candidate sequences. For example, the realization reranking strategy in Figure 6(b) makes use of parallel back-off to learn patterns such as *serves X food* or *is an X restaurant*. This can be achieved by allowing the probability of a phrase to depend on the phrase at time $t-2$ rather than on the preceding phrase (see right branch in Figure 6(b)). Hence if the pattern exists in the training data, $p(r_t|l_{t-1}, r_{t-2})$ is likely to be preferred over $p(l_{t-1}, r_{t-1})$ during reranking, for example, giving a large probability of r_t ='*food*' if r_{t-2} ='*serves*' and l_{t-1} =`inform(food(SOMETHING))`.

5. Stochastic Paraphrase Generation

Because a dialogue act can typically be conveyed in a large number of ways, it seems natural to model the NLG task as a one-to-many mapping. However, previous work on statistical NLG has typically focused on evaluating the top ranked utterance, without evaluating whether the generator can produce paraphrases matching a reference paraphrase set (Langkilde-Geary 2002; Reiter and Belz 2009). Although single-output NLG is acceptable for one-off text generation, NLG systems used within long-term human–computer interaction are likely to benefit from modeling the paraphrasal variation found in human language (e.g., by reducing the repetitiveness of dialogue system utterances or by improving the chances of successful dialogue clarifications).

However, learning to map a single input to a *set* of surface realizations is a nontrivial machine learning problem. One advantage of casting the NLG task as search over FLMs is that the final n-best list of surface realizations can be used to constrain the search for valid paraphrases. See Table 2 for examples of BAGEL's n-best outputs in the

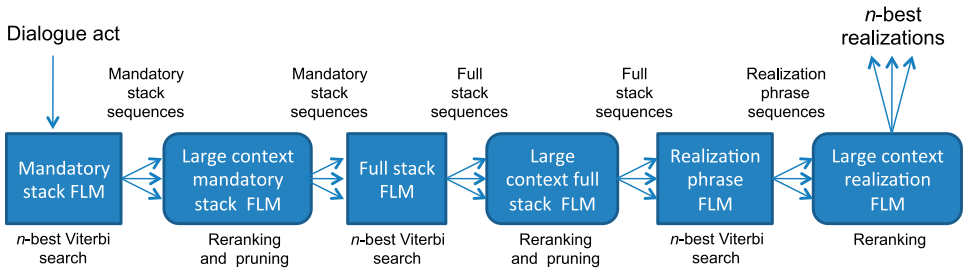
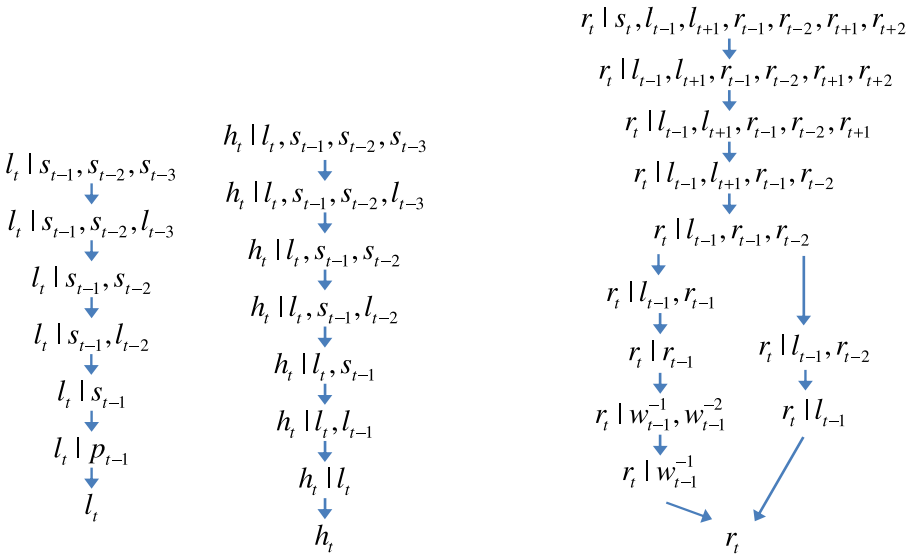


Figure 5
Architecture of an FLM-based statistical generator using cascaded large context reranking FLMs.



(a) Semantic stack head h_t and tail l_t given previous stacks $s_{t-1}, s_{t-2}, s_{t-3}$. The full stack probability is obtained by multiplying both factors using the chain rule.

(b) Realization phrase r_t given surrounding phrases $r_{t-1}, r_{t-2}, r_{t+1}, r_{t+2}$, semantic context s_t, l_{t-1}, l_{t+1}, and preceding words $w_{t-1}^{-1}, w_{t-1}^{-2}$.

Figure 6
Back-off graphs for (a) both semantic stack reranking FLMs and (b) the realization phrase reranking FLM.

tourist information domain. This section proposes two methods using those outputs to generate paraphrases that can be used interchangeably in dialogue.

5.1 *n*-best Selection Beam for Paraphrasing

We first propose taking a sample from the top of the *n*-best list produced by BAGEL's realization reranking FLM shown in Table 2. However, to avoid sampling from the long tail of low-probability utterances, we only consider utterances whose probability lies within a selection beam relative to the probability first-best utterance p_1; that is, only the utterances generated with a probability above

$$p_{\texttt{min}} = p_1 \cdot (1 - \texttt{selection beam})$$

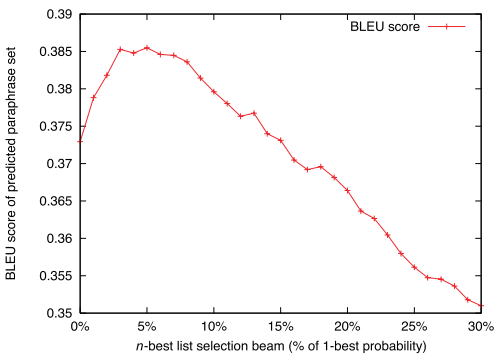
are kept. The top utterances are typically grammatical and natural; however, determining a cut-off threshold that captures some of the linguistic variation found in the data without introducing disfluencies is a nontrivial problem. Because many system acts are associated with multiple reference paraphrases in our data, the BLEU score (Papineni et al. 2002) can be used to tune the threshold value. BLEU is a corpus-level metric that is typically used to evaluate a test corpus against a set of reference paraphrases. In order to evaluate the worth of the predicted *set* of utterances, each utterance within the selection beam is considered as part of the test corpus, thus favoring models generating multiple utterances matching any of the reference paraphrases rather than a single utterance. Figure 7(a) shows the BLEU score of paraphrase sets generated using different *n*-best selection beams, averaged over a 10-fold cross-validation over 1,646 distinct dialogue

Table 2
Example *n*-best lists produced by BAGEL with FLM reranking (after normalizing the probabilities to 1, but before thresholding).

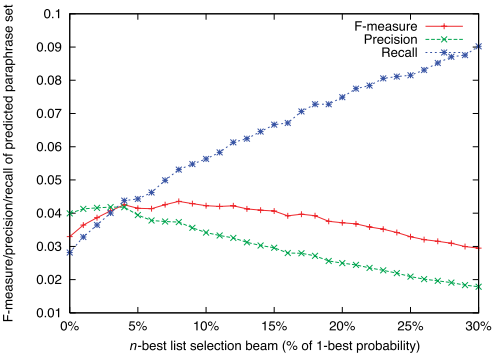
***n*-best list**	**Prob**
`inform(name(X) area(centre) food(Y))`	
X serves Y food in the city centre.	0.044
X is an Y restaurant in the city centre.	0.035
X serves Y food in the centre of town.	0.033
X serves Y food in the centre of the city.	0.033
X is a Y restaurant in the city centre.	0.029
X is a Y food in the city centre.	0.028
`inform(name(X) area(centre) seetype(architecture))`	
X is an architectural building in the city centre area.	0.025
X is an architectural building in the city centre.	0.024
X is an architectural building. It is located in the centre of the city.	0.022
X is an architectural building in the centre of town.	0.022
X is an architectural building in the centre of the city.	0.022
X is an architectural building. It is located in the city centre.	0.020
`request(area)`	
Whereabouts were you thinking of?	0.141
In which area of town would you like to eat?	0.136
What type of area are you looking for?	0.020
What type of area would you like?	0.020
What kind of? Area are you looking for?	0.019
Whereabouts are you looking for?	0.018
`reject(near(X) unitype(department))`	
There are no university departments near X.	0.091
Unfortunately, there are no university departments near X.	0.031
I'm sorry, there are no university departments near X.	0.028
Unfortunately, there are no, there are no university departments near X.	0.026
I'm sorry, there are no, there are no university departments near X.	0.024
I am sorry, there are no university departments near X.	0.023
I'm sorry, but there are no, there are no university departments near X.	0.020

act and paraphrase set pairs collected through crowdsourcing. The data collection process is detailed in Section 6. It is important to note that none of the dialogue acts used for testing were seen at training time. The BLEU score was computed by treating all predicted paraphrases as a whole document. We find that including the top 6% of the *n*-best list produces a higher BLEU score than using the first-best utterance only (BLEU = .39 vs .37). As a high level of overlap with a reference utterance does necessarily result in grammatical or natural outputs, Figure 7(b) also looks at the precision and recall of the generated paraphrase set given the reference set (i.e., only considering *exact* utterance matches). Although exact matches are rare on unseen inputs, we find that the optimal F-measure is obtained when considering the top 8% of the probability mass of the *n*-best list, which corresponds to an average of 2.1 paraphrases, according to Figure 8. Both evaluation metrics suggest that generating paraphrases improves linguistic variation without affecting grammaticality, hence potentially improving naturalness in dialogue. Unless stated otherwise, we use a selection beam of 8% in our experiments.

Table 2 also provides some insight into the potential limitations leading to unnatural outputs. We find that some errors arise from the separation between the semantic stack decoding step and the realization step, together with an excess of smoothing. For



(a) Average BLEU score of the predicted paraphrase sets



(b) Precision, recall, and F-measure (exact match)

Figure 7
Automated evaluation of BAGEL's predicted paraphrase sets for different *n*-best selection beams. Results are averaged over a 10-fold cross-validation.

example, *X is a Y food in the city centre* in the first section of Table 2 was associated with a non-zero probability because the phrase sequence *X serves Y food* occurs frequently in the data, hence allowing the stack `inform(food(Y))` to be followed by `inform(food)` rather than `inform(type(restaurant))`. At the realization stage, the *is a* realization phrase is associated with a high probability, given an `inform` stack following a restaurant name and a sentence start symbol, while the phrase *food* following *is a Y* is allowed because the unseen context gets dropped by the back-off strategy. Similarly, the example *unfortunately, there are no, there are no university departments near X* in the last section of Table 2 is associated with a non-zero probability because the semantic stack decoding step predicted multiple `reject` stacks followed by a punctuation mark because the non-adjacent stack context was smoothed away, leading to phrase repetitions at the realization stage. Although these type of errors are typical of sequential models trained on a limited amount of data, they tend to be associated with a lower probability than the top hypotheses, and additional data would make such errors less likely by allowing for

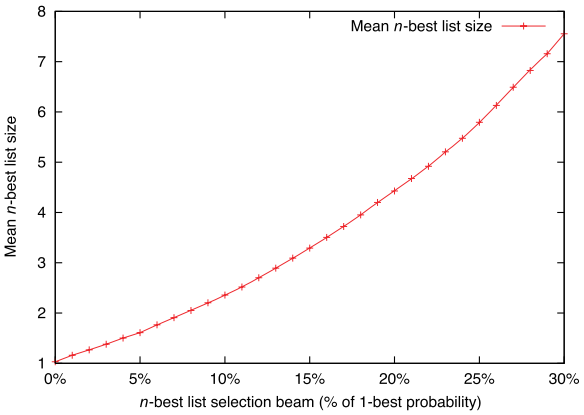


Figure 8
Mean size of the predicted paraphrase sets for different FLM selection beams. Results are averaged over a 10-fold cross-validation.

larger contextual dependencies to be modeled without back off. However, FLMs will always associate a small probability to a large range of utterances; hence there is a need for selecting paraphrases based on a selection beam or statistical classification methods.

5.2 Structured Perceptrons for Paraphrase Classification

FLMs can be trained easily by estimating conditional probabilities from feature counts over a corpus, and they offer efficient decoding techniques for real-time generation. However, FLMs do not scale well to large feature sets (i.e., contexts), as each additional feature increases the amount of data required to accurately estimate the FLM's conditional probability distribution. Backing off as described in Section 4.2 alleviates this issue, although finding the optimal back-off strategy is nontrivial even for small feature sets (e.g., 10 features). Furthermore, FLMs are trained to maximize the likelihood of the training data; hence utterances containing frequent phrases are more likely to be generated than utterances containing infrequent phrases, even if the latter is part of the training set. Whereas in the previous section, a selection beam was optimized for selecting paraphrases, it is learned once and for all regardless of the input. This section therefore investigates whether performance can be improved through discriminative training, by rescoring the list of candidate semantic stack and realization sequences produced by the FLMs based on *binary classification models* predicting whether each candidate sequence is a valid paraphrase. We propose a training method inspired by Collins' work on discriminative reranking for part-of-speech tagging and syntactic parsing, which uses the *structured perceptron* on-line algorithm to learn to rerank the output of a generatively trained model (Collins 2002a, 2002b; Collins and Roark 2004). The structured perceptron algorithm learns a linear discriminant function of the features $\Phi(x, y)$ of both the input structure x and the output structure y (e.g., semantic stack and realization phrase sequences, respectively) by iteratively updating its feature weights α each time it wrongly predicts a training example. Each update makes the weight vector closer to the features of the training example, and further away from the incorrect prediction. A crucial point is that each prediction requires finding the output z that maximizes the discriminant function given the input x. As a Viterbi search is not tractable because of the large context dependencies of the features, we limit our search to sequences in the n-best list $GEN(x)$ produced by the short context FLMs.

Although structured perceptrons were previously used to learn a reranking function (Collins 2002a; Collins and Roark 2004), the resulting scores cannot be used directly to select multiple valid paraphrases among the candidates. Rather than learning a cut-off threshold as done in Section 5.1, we cast the perceptron reranking step as a binary classification task, by updating the perceptron's weight vector accordingly each time (a) a reference realization is classified negatively and (b) a non-reference realization in $GEN(x)$ is classified positively. The main difference with Collins' reranking model is that the zero of the discriminant function is trained to act as a classification threshold. At generation time, the learned model classifies each candidate realization of $GEN(x)$ to determine whether it should be included in the paraphrase set from which the final utterance can be selected. It is important to note that this approach iterates over training pairs generated from the same input dialogue act. A consequence is that the data is no longer independently and identically distributed, thus potentially increasing the generalization error of the models.

The resulting *kernelized structured perceptron* algorithm adapted to our task is given in Table 3, which learns a set of feature vectors and their corresponding weights. To facilitate understanding, Table 3 also presents the simplified algorithm in the case of a

Table 3
The generic kernelized perceptron training algorithm for structured prediction, as well as the simplified version using a linear kernel. Both algorithms were adapted to the NLG reranking task.

Input: T training iterations, n training examples associating each input x_i with an output set $\mathcal{Y}_i$ (i.e., semantic stack or realization sequences). $GEN(x_i)$ returns the n-best output sequences for input x_i based on a Viterbi search using the corresponding FLM, in which n depends on a pruning beam and a maximum value. $\Phi(x_i, y)$ is a sparse feature vector of dimensionality d representing the number of occurrences of specific combinations of realization phrases and/or semantic stacks in (x_i, y), with an entry for each instantiation in the training data of each node of the backoff graph of the large context FLM in Figure 6.

Output: a collection V of feature vectors in $\mathbb{R}^d$ and their respective weights α in $\mathbb{R}^{|V|}$. Using a linear kernel, the algorithm is simplified as the weighted feature vectors can be represented as a single weight vector $w = \sum_{j=1}^{|V|} \alpha_j V_j$ in $\mathbb{R}^d$.

Linear kernel algorithm:
$w = \vec{0}$
For $t = 1...T, i = 1...n$
  For z in $GEN(x_i) - \mathcal{Y}_i$
    If $w.\Phi(x_i, z) \geq 0$ then $w \leftarrow w - \Phi(x_i, z)$ // incorrect positive prediction
  For y in $\mathcal{Y}_i$
    If $w.\Phi(x_i, y) < 0$ then $w \leftarrow w + \Phi(x_i, y)$ // incorrect negative prediction

Kernelized algorithm with kernel function $K : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$**:**
$V = [\vec{0}] \quad \alpha = [0]$
For $t = 1...T, i = 1...n$
  For z in $GEN(x_i) - \mathcal{Y}_i$
    If $\sum_{j=1}^{|V|} \alpha_j K(\Phi(x_i, z), V_j) \geq 0$ then // incorrect positive prediction
      append $\Phi(x_i, z)$ to V // weigh instance negatively
      append -1 to α
  For y in $\mathcal{Y}_i$
    If $\sum_{j=1}^{|V|} \alpha_j K(\Phi(x_i, y), V_j) < 0$ then // incorrect negative prediction
      append $\Phi(x_i, y)$ to V // weigh instance positively
      append 1 to α

linear kernel, in which the weighted feature vectors are collapsed into a single weight vector. In our experiments, we use a polynomial kernel of degree 3. The feature vectors represent the number of occurrences of specific combinations of realization phrases and/or semantic stacks in the input and output sequences, with an entry for each instantiation in the training data of each node of the back-off graph of the large context FLM in Figure 6. For example, the back-off node $r_t|l_{t-1}, r_{t-2}$ in Figure 6(b) is used to derive a feature characterizing the number of occurrences of the phrase *has* followed by the stack tail `inform(food(SOMETHING))` followed by the phrase *food*.

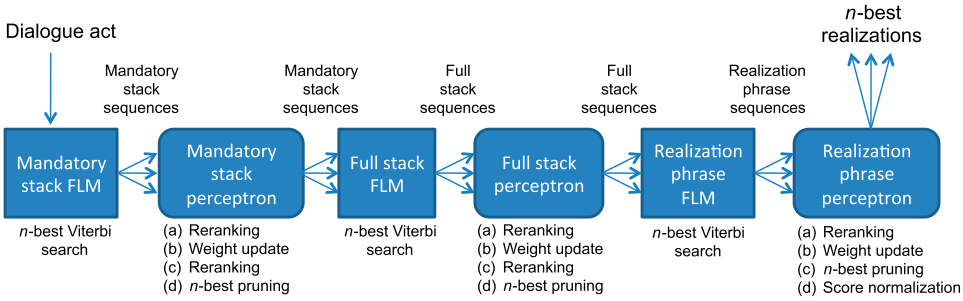


Figure 9
Online discriminative training of an FLM-based statistical generator using cascaded reranking perceptrons. The only differences between the training and generation process is that (a) weights are not updated at generation time and (b) only one reranking step is performed after each decoding stage.

Rather than using the final weight vector to make predictions at generation time, using the *averaged* weight vector over all updates was shown to generalize better to unseen examples (Collins 2002a). Collins has shown that structured perceptrons can outperform boosting and SVM-based models, with a training complexity growing linearly with the training set size (as opposed to a cubic complexity for large-margin classifiers).

The resulting NLG pipeline is illustrated in Figure 9, with a perceptron model reranking the output of each FLM decoding model. All perceptron models are trained simultaneously by iteratively generating each training example, and updating each reranking model if its first-best sequence differs from the reference sequence. This results in three instantiations of the perceptron algorithm in Table 3. As the output of the first model in the pipeline affects the training process of subsequent models, the candidate *n*-best list is reranked twice: (a) before updating the perceptron's weight vector in order to find whether the current best hypothesis matches the reference, and (b) after updating the weight vector to maximize the accuracy of the input to subsequent models in the pipeline.

6. Corpus Collection

Our target domain is a large-scale spoken tourist information system for Cambridge. Table 4 illustrates the 10 types of dialogue acts that are produced by the dialogue manager. Because each dialogue act type exhibits different stack ordering patterns, they require distinct semantic stack prediction models. Some of the communicative goals include logical operators, such as global negations and logical quantifiers (e.g., rows 2, 4, and 5 in Table 4), each of which require a specific dialogue act type. Figures 10[3] and 11 illustrate the ontology of our domain, which results in 128 distinct semantic stack concepts (e.g., characterizing whether a venue is a bar, museum, cinema, but also whether it is cheap, near another venue, whether it has Internet, parking space, or whether it allows children). Because our approach targets dialogue applications, BAGEL's semantic representation is defined by the domain ontology itself; hence the semantic concepts typically correspond to constraints used to narrow down the user goal. In information presentation systems, such concepts are typically associated with database attributes of

3 This figure is included to indicate the complexity of the ontology; it is not intended that the detail be legible.

Table 4
Example realizations for the 10 dialogue act types supported by BAGEL.

#	Dialogue act and first-best realization
1.a	`confirm(eattype(pub) hasfood)` *To confirm, you are looking for a pub that serves food?*
1.b	`confirm(dontcare(area) drinktype(bar))` *Could you confirm that you want a bar in any area?*
2.a	`reject(area(X) sport(gym) type(sportsvenue))` *There are no sports centres with a gym in X.*
2.b	`reject(known(hasinternet) near(X) type(placetodrink))` *Unfortunately, there are no bars for which i have information about the internet near X.*
3	`inform(name(X) unknown(childrenallowed) drinktype(bar) hasmusic unknown(hastv))` *X is a bar with music, but there is no information on whether children are allowed or whether it has television.*
4	`informall(all(eattype(restaurant) food(X)) area(riverside))` *All the restaurants serving X food are in the riverside area.*
5	`informonly(only(name(X)) eattype(restaurant) near(Y) pricerange(moderate))` *X is the only moderately priced restaurant near Y.*
6	`suggest(pricerange(cheap) pricerange(expensive) pricerange(moderate))` *For example, you can have somewhere cheap, expensive, or somewhere in the middle.*
7.a	`request(pricerange)` *What kind of price range would you like?*
7.b	`request(hasparking)` *Would you like a place with a parking space?*
8.a	`select(type(placetodrink) type(placetosee))` *Do you want to find somewhere to get a drink or go and find somewhere to see?*
8.b	`select(area(X) dontcare(area))` *Would you like that to be in the X area or would you like me to search all areas?*
9	`repeat()` *Could you please repeat that?*
10	`reqmore()` *Can I help you with anything else?*

the entities of interest. In our framework, the ontology is shared between the dialogue manager, the language understanding component, and the NLG component. Our ontology was thus refined over a long period of time prior to this work. The manual effort required for defining an ontology for a new domain is highly dependent on the domain granularity. While automatically deriving ontologies for complex domains remains an unsolved problem, in recent work an ontology for a bus transportation dialogue system was handcrafted in a matter of days (Thomson et al. 2010).

Because there is no feedback between the language generator and the dialogue manager, the NLG component is expected to handle any combination of dialogue act type and semantic concept arguments. The main advantage of data-driven methods over handcrafted methods is their potential for scaling to such large domains by shifting the bulk of the development effort from manual tuning to data collection. However, a major issue is that such methods typically require semantically annotated data, which is costly to collect. Furthermore, domain data is rarely available; hence a creative process is required for generating a wide range of domain utterances. This article is based on the assumption that learning to produce paraphrases can be facilitated by collecting data

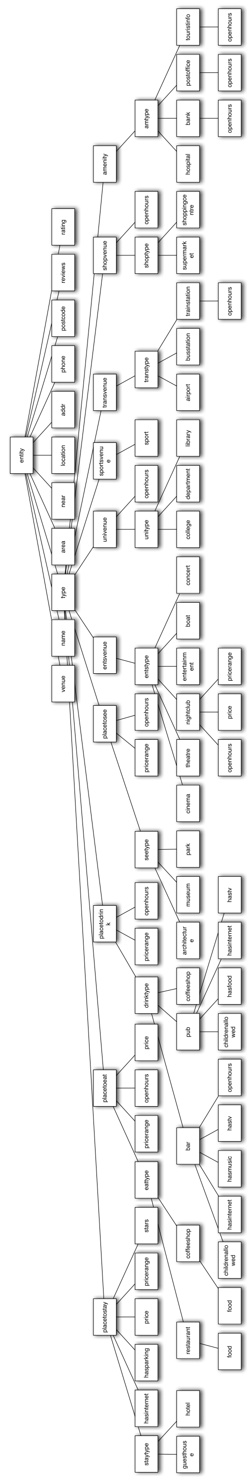


Figure 10
Full ontology of the Cambridge Tourist Information System (CamInfo).

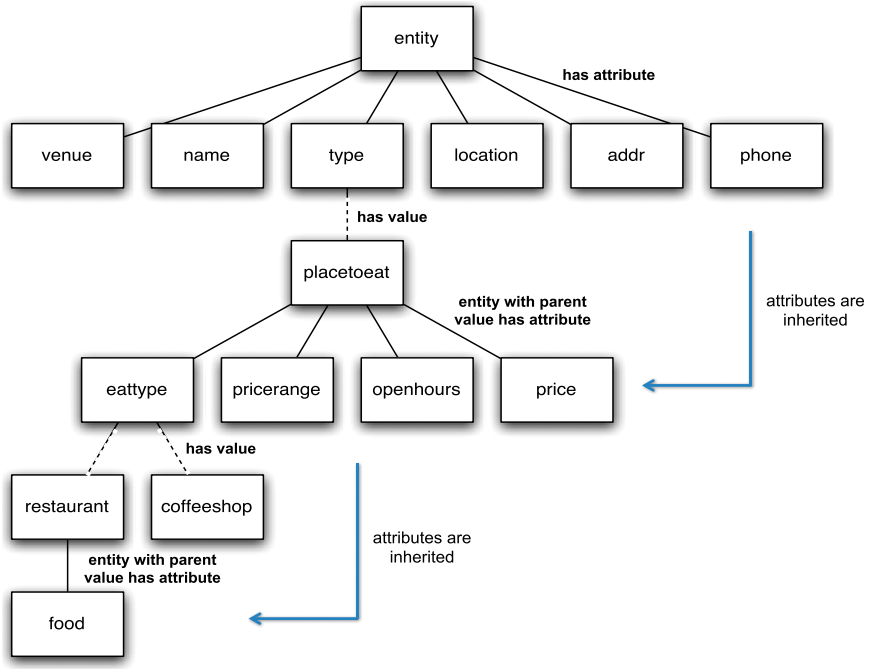


Figure 11
Partial ontology for places to eat. All relations are pointing downwards. Attributes at higher level are inherited for entities matching specific attribute values (dashed lines); for example, all entities with attribute `eattype` set to `restaurant` have the attributes `food`, `price`, `phone`, and so on.

from a large sample of annotators. However, this requires that the meaning representation should be simple enough to be understood by *untrained* annotators. This section describes how we make use of BAGEL's coarse-grained semantics to collect data from a large sample of untrained annotators, using Amazon's Mechanical Turk.

A crucial aspect of data collection for NLG is to ensure that the annotators understand the meaning of the semantics to be conveyed. Annotators were first asked to provide an utterance matching an abstract description of the dialogue act, regardless of the order in which the constraints are presented (e.g., *Offer the venue Taj Mahal and provide the information type(restaurant), area(riverside), food(Indian), near(The Red Lion)*). The order of the constraints in the description was randomized to reduce the effect of priming. The annotators were then asked to align the attributes (e.g., *Indicate the region of the utterance related to the concept* 'area'), and the attribute values (e.g., *Indicate only the words related to the concept* 'riverside'). The total input semantic space is approximated by the set of system dialogue acts produced during 250,000 simulated dialogues between our statistical dialogue manager (Young et al. 2010) and an agenda-based user simulator (Schatzmann et al. 2007). In order to build the training set, we started with a set of utterances collected for a small subset of our domain (Mairesse et al. 2010). We then ordered the dialogue acts based on their frequency of occurrence in the simulated dialogues. In order to ensure that each semantic stack defined by the domain ontology occurs at least once in our data, we expanded our training set by iteratively adding the most frequent unseen act which contains an unseen mandatory semantic stack. The resulting data set consists of 1,646 unique dialogue acts after replacing non-enumerable values by a generic symbol. Each dialogue act contains an average of 3.27

mandatory semantic stacks. We generally collected one utterance per act, although two paraphrases per act were collected during our initial experiment. The resulting data set contains a total of 1,956 aligned utterances produced by 137 native speakers of English. After manually checking and normalizing the data set,[4] the layered annotations were automatically mapped to phrase-level semantic stacks by splitting the utterance into phrases at annotation boundaries. Each annotated utterance is then converted into a sequence of symbols such as in Table 1, which are used to estimate the conditional probability distributions defined in figures 4 and 6. The resulting vocabulary consists of 864 distinct semantic stacks and 1,180 distinct realization phrases, with an average of 7.35 phrase/stack pairs per utterance.

7. Evaluation

This section evaluates BAGEL in the tourist information domain, using an automated metric as well as human judgments of resynthesized dialogues. Our objective is not only to evaluate the naturalness of the generated utterances for different training methods, but also to assess whether the linguistic variation found in BAGEL's outputs improves the naturalness of the overall dialogue interaction.

7.1 Comparison with Utterance Class Language Models

As Oh and Rudnicky's LM-based approach is the first statistical NLG method that requires almost no handcrafting (Oh and Rudnicky 2002), we first compare their method to BAGEL in our domain and discuss the differences between both approaches.

7.1.1 Utterance Class LM Baseline. Oh and Rudnicky's (O&R) approach trains a set of word-based *n*-gram language models (LMs) after replacing slot values by placeholder variables. In order to bias the LMs towards specific intents, the LMs are trained on subsets of the data referred to as **utterance classes**. An utterance class is the set of utterances matching a specific dialogue act type and a set of zero or more slots. For example, `inform(near(X))` would be a valid utterance class, characterizing all the utterances with the `inform` dialogue act type and at least one `near` slot. Given large domains, evaluating all possible utterance class partitions of the data is not tractable: In their experiments in the air travel domain, O&R limit their utterance classes to at most one slot. In order to identify how to partition our data, we investigate a number of utterance classes: (a) using dialogue act types only; and (b) including one or more slots. Because deciding what slot to include is a nontrivial problem, we include slots based on their frequency of occurrence in the utterance class. The utterance class `nomatch(eattype(restaurant) near(X))` for instance indicates that `eattype(restaurant)` and `near(X)` are the two most frequent slots for the dialogue act type. Note that such an utterance class can also generate other slots besides those belonging to the class, the main difference being that those other slots act as run-time constraints in the overgeneration phase, whereas utterance class slots constrain the model's training data.

At generation time, the LM for the utterance class matching the input is used to overgenerate a set of candidate utterances in a depth-first fashion by sampling from the LM distribution, one word after the other. Because BAGEL relies on the prediction of an

4 The manual verification took around 15 person-hours for 1,956 utterances. It involved correcting English disfluencies and semantic misinterpretations. No samples were deleted.

end symbol, we extend O&R's model with an end symbol determining when to end the utterance. In addition to random sampling, we also implemented a deterministic version of the algorithm that generates all words that followed the utterance context in the training data, as long as they do not violate input constraints (i.e., generate unspecified slots). Decoding was halted if the utterance generated more than 20 words. Although it was not needed on our data set, it is important to note that such a greedy search is likely to require beam pruning on larger data sets. We find that the deterministic version both improves performance and makes it more comparable with BAGEL's decoding algorithm. Additionally, in order to investigate the effect of the granularity of emission symbols on performance, we also train a *phrase-based* version of the baseline in which the LMs are trained to predict symbols representing contiguous words either within or between surface slots. In all baselines, the final utterance is selected based on an implementation of the rescoring rules used in O&R, which rescore the utterance based on whether:

1. The utterance is too short or too long. The probability of the generated utterance is weighted by the probability of the utterance length given the utterance class according to the training data.
2. The utterance contains repetitions of any of the slots.
3. The utterance contains slots for which there is no valid value in the input.
4. The utterance lacks any of the required slots.

The last three rules result in a multiplicative weight of 10^{-9}, that is, the utterance would only be chosen if no other candidates satisfy the slot constraints. The system returns the most highly scored utterance over 10,000 iterations for the sampling baseline (vs. 50 in O&R's experiments). Additionally, our implementation of O&R's method keeps track of visited slots during generation, hence pruning paths that generate a slot placeholder which is not part of the input, or generate a slot more times than specified in the input.

We train models on the same 10-fold cross-validation folds as in Section 5.1, namely, partitioning the 1,646 distinct dialogue acts for which we collected one or more utterances. None of the test dialogue acts are present in the training folds. Results report the BLEU scores averaged over the 10 test folds.

7.1.2 Results. A first result shown in Table 5 is that O&R's original sampling approach does not perform as well as the deterministic algorithm, while being more computationally expensive. A paired t-test over the 10 cross-validation folds reveals that the difference is significant for all configurations ($p < 0.01$ or lower, two-tailed). The sampling size used is much larger than in O&R's experiment, suggesting that sampling does not scale well to larger domains. The rest of this section refers to the deterministic approach.

We also find that none of the *phrase-based* O&R models produce BLEU scores above .10. We believe this is due to the lack of semantic labels besides slot values, which causes phrases to be very long and unlikely to occur both in the training and test folds. The rest of this section therefore refers to O&R's word-based approach.

Table 5 shows that on our data set O&R's method is sensitive to the granularity of the utterance class. The trigram model performs best without including any slot in the utterance class, with a mean BLEU score of .28. In contrast, BAGEL produces a score of .37 on the same data (using the most likely utterance only). A paired t-test shows that this score is significantly higher (two-tailed, $p < 0.0001$). The configuration

Table 5
BLEU score of the word-based utterance class LMs for different *n*-gram sizes and different number of slots included in the utterance class (most frequent first). Best performing parameters are in **bold**. The BLEU score is averaged over all cross-validation folds. See figures 12 and 13 for results using other parameter configurations.

System configuration	*n*-gram size	BLEU no slot	BLEU 1 slot	BLEU 2 slots
O&R deterministic	2	.25	.06	.05
O&R deterministic	3	**.28**	.02	.01
O&R deterministic	4	.25	.01	.00
O&R sampling	2	.25	.03	.03
O&R sampling	3	**.27**	.02	.01
O&R sampling	4	.23	.01	.00
Bagel	n/a		**.37**	

in which the utterance class consists of the dialogue act type only (i.e., no slots) is the only one producing an output utterance for almost all unseen inputs in the test folds (99% for bigram LMs, 93% for trigram). Figure 12 illustrates results for additional slot combinations, showing that adding more slots consistently decreases performance. Figure 13 shows that this performance decrease can also be observed when using sampling.

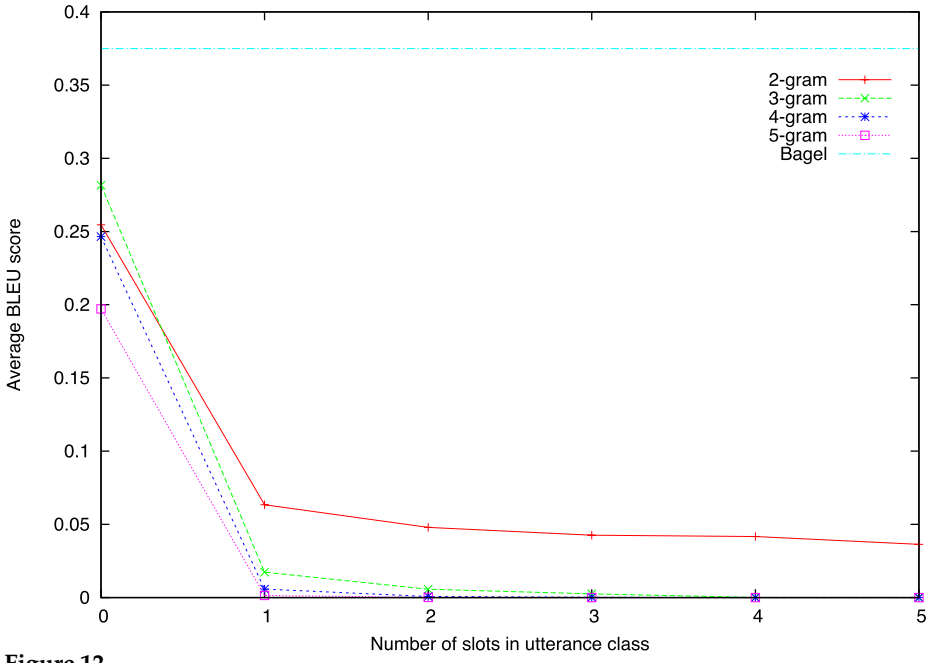


Figure 12
BLEU score of the word-based utterance class LMs with deterministic decoding for different *n*-gram sizes and different number of slots included in the utterance class (most frequent first). The BLEU score is averaged over all cross-validation folds. *Bagel* indicates the best performing BAGEL configuration on the same folds.

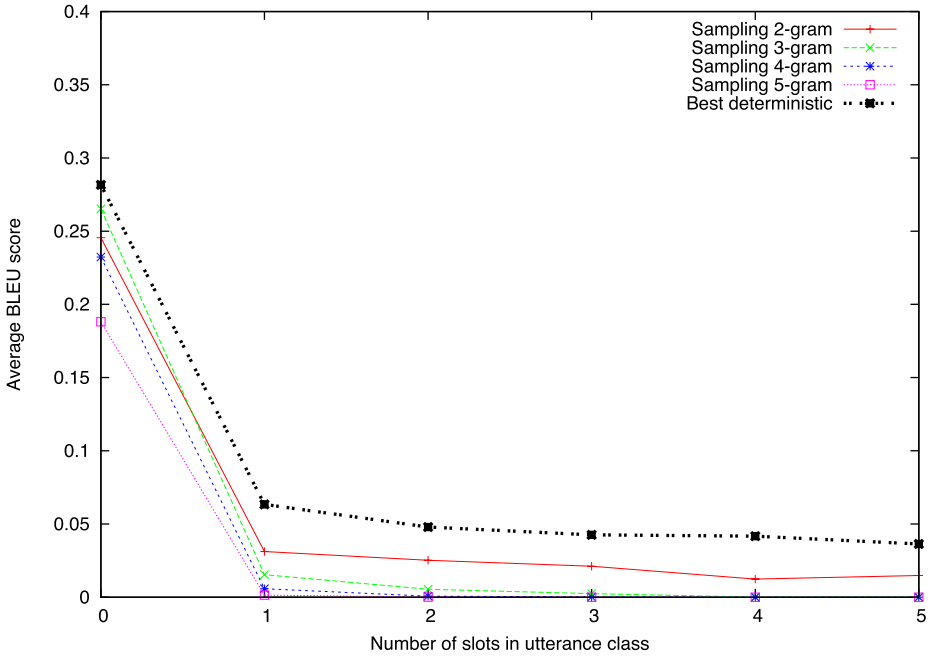


Figure 13
BLEU score of the word-based utterance class LMs with sampling for different n-gram sizes and different number of slots included in the utterance class (most frequent first). The BLEU score is averaged over all cross-validation folds. The scores obtained with the best deterministic version are included for comparison.

We find that the addition of the most frequent slot to the utterance class decreases performance significantly with a BLEU score of .06 with a bigram model and .02 with a trigram model ($p < 0.0001$ for both, two-tailed). Figure 12 suggests that performance decreases further with larger n-gram sizes. This decrease is likely to be due to the fragmentation of the training data illustrated in Figure 14, as sparser probability counts make the generation process less likely to find a path satisfying the global slot constraints. For instance, adding the most frequent slot in the training data as part of the utterance class causes more than half of the test input to produce no prediction using a bigram model. Although removing the decoding constraints is not tractable, we can estimate the performance of O&R's method given unlimited computing power by only evaluating it on the subset of the data for which the constraints are not violated—that is, on the test data which does produce an output utterance. In this case the best O&R baseline yields a score of .32 on successful predictions (69% of the data) using the 5-gram model with no slots, whereas the same model yields a score of .20 when taking all test utterances into account.

A general issue is that although a broad utterance class reduces data sparsity, it learns a model more likely to produce the most frequent patterns in the utterance class, making it difficult to model specific slot combinations correctly. An utterance class including many slots can model those slots more accurately; however, it can only be trained on the fraction of the data matching that class, creating data sparsity issues.

Regardless of the utterance class size, we find that O&R's baseline performance decreases for contexts larger than trigrams. For example, Figure 12 shows that the BLEU score decreases significantly from .28 for trigrams to .24 and .20 for 4-grams and

5-grams, respectively ($p < 0.0001$ for all differences, no slots in the utterance class). This decrease in performance is likely to be due to overfitting. Larger *n*-grams are less fertile because they result in fewer non-zero transitions from a given context; hence they are less likely to produce an utterance satisfying the slot constraints. This particular issue could be alleviated by investigating different smoothing strategies.

Given the large differences in BLEU score observed and the limited resources available, we did not evaluate O&R's approach using human judges. It is important to note that a human evaluation would be desirable to strengthen our findings. Additionally, future work should evaluate whether the difference in performance holds for larger data sets.

7.1.3 Discussion. Like BAGEL, O&R's method uses a search over a sequential probabilistic model of a phrase given its context. However, a major difference with our approach is that semantic concepts are only explicitly modeled through slot placeholders and the utterance class. A limitation is therefore that it requires the definition of an optimal utterance class partition before training, namely, determining what slots the words should be conditioned on, if any. Including all slots as part of the utterance class would highly fragment the data, whereas using only the dialogue act type is likely to reduce the model's capability of producing slot-specific phrasings. As shown in our experiments, the choice of what slots to include in the utterance class has a large impact on the quality of the output utterances. BAGEL mitigates this by not conditioning the generated words on a global utterance class value, but by conditioning the individual words on elements of a generated sequence of semantic symbols. Given that the number of semantic concepts is lower than the vocabulary size, using an explicit semantic representation can reduce the number of parameters to estimate during training compared with systems relying on various word contexts. In some cases, however, the previous words provide

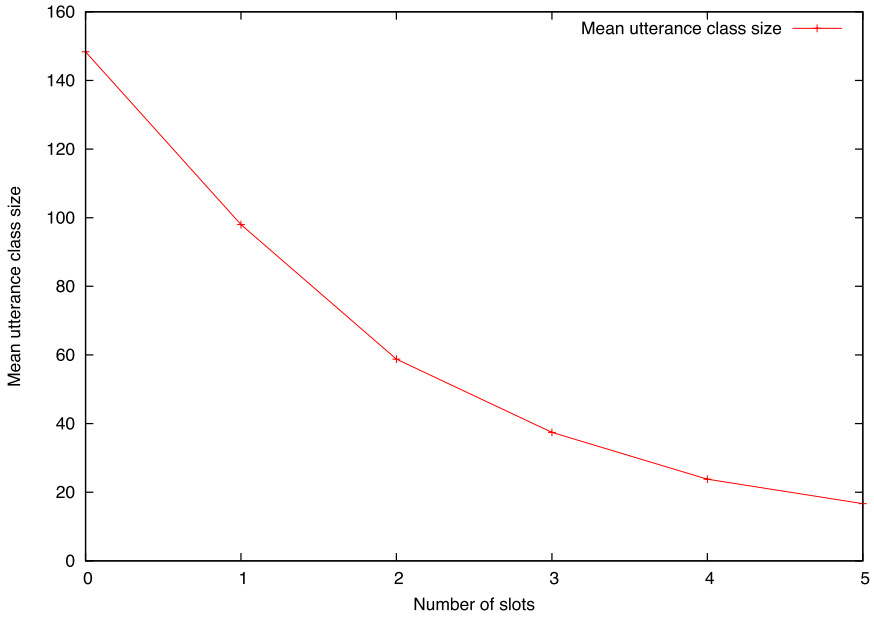


Figure 14
Mean number of training utterances per utterance class for different number of slots included in the class (most frequent slot first). Without any slot the utterance class consists of the dialogue act type.

additional useful information (e.g., for local agreement); hence there is value in taking both the semantic and word context into account whenever needed. Factored language models provide a way for the learner to choose what context to rely on.

Finally, another difference with our approach is that the lack of hierarchical semantics implies that each lexical item realizing an input slot value has to be specified in the input. This is a valid approach for domains in which slot values are limited to numerical values or proper nouns, but not for domains in which semantic concepts need to be realized differently, depending on the context and the dialogue act type. For example, compare how BAGEL realizes the `area` semantic concept in the query *Whereabouts were you thinking of?* as opposed to in the statement *Char Sue is located in the Arbury area*. Requiring each slot value to be realized using the same lexical item regardless of the context is likely to be impossible for large domains, especially with multiple dialogue act types. This limitation could be alleviated by including the n slots for which we want to control the lexical realization as part of the utterance class. However, this is not tractable as it would require fragmenting the data further to produce all 2^n slot combinations as distinct utterance classes. Sharing data across utterance classes or using hierarchical class-based language models could mitigate this issue, but this is beyond the scope of this article.

This section has shown that BAGEL's FLM approach significantly outperforms utterance class-based LM methods on our data using automated evaluation metrics. We now evaluate BAGEL using human judgments.

7.2 Human Evaluation from Text Samples

Although automated metrics provide useful information for tuning model parameters, they only correlate moderately with human naturalness ratings (Papineni et al. 2002). We therefore evaluate the methods presented in the previous sections through a subjective rating experiment, using Amazon's Mechanical Turk services. For each dialogue act in our unseen test set, we generate a set of paraphrases with each of the following system configurations: (a) using large context reranking FLMs (*FLM*); (b) using perceptron reranking (*perceptron*); and (c) using the output of the decoding models directly (*no reranking*). In order to validate the paraphrasing FLM threshold analysis presented in Section 5.1, we evaluate utterances generated within a selection beam of 8% and 15% relative to the probability of the top hypothesis (FLM_8 and FLM_{15}), as well as a system returning the top hypothesis only (FLM_0). For each configuration, we either train all decoding and reranking models on distinct data sets for each dialogue act type in Table 4, or we train a single realization model on all dialogue act types (*global*). Although a global realization model can potentially generalize across dialogue act types (e.g., not requiring each top semantic concept to be seen with each act type during training), performance is likely to be affected by the resulting increase in vocabulary size and the reduction in consistency between training examples.

Concerning the perceptron reranking algorithm, we use a kernelized perceptron with a polynomial kernel of degree 3 as it performed best in preliminary experiments on a subset of our training data. We evaluate all the paraphrases classified as positive by the model for a given input act. Our experiment compares two variants of the perceptron model: (a) using the weights of the last perceptron update (*Last*); and (b) taking the average of each weight update weighted by the number of instances for which the weight vector was left unchanged during training (*Avg*). In order to account

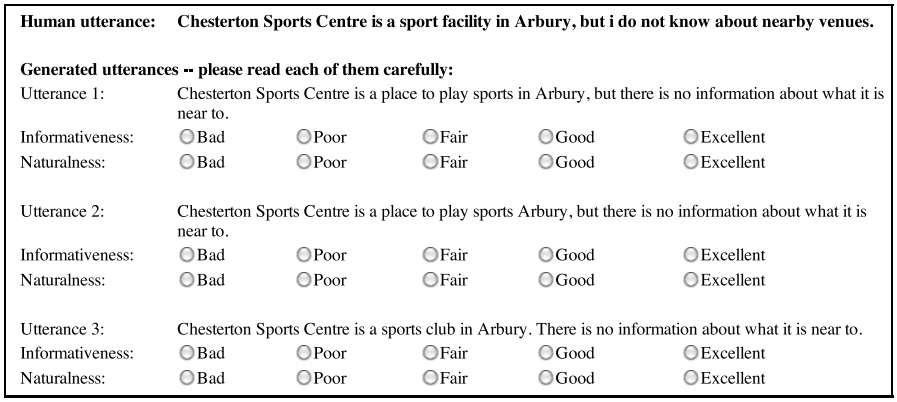


Figure 15
Human evaluation interface for text-based utterance evaluation. The generated utterances are presented in random order.

for differences in computational resources needed by each system, we set the pruning thresholds such that each paraphrase set is generated within 0.5 seconds on a Pentium 4.2 GHz. For each input dialogue act, a maximum of 100 realizations were reranked in our experiments. These were derived from up to five semantic stack sequences, each generating up to 20 realization phrase sequences.

For the purpose of the evaluation, the generated paraphrase sets for all systems are combined and presented in random order, for four dialogue acts at a time. Participants were told that each utterance was meant to have the same meaning, and they were asked to evaluate their *naturalness* on a 5-point Likert scale, as illustrated in Figure 15. Naturalness is defined as whether the utterance could have been produced by a human. Each utterance is taken from the test folds of the cross-validation experiment presented in Section 5.1—that is, the models are trained on up to 90% of the data and the training set does not contain any of the generated dialogue acts.

7.2.1 Results. Table 6 presents the average naturalness rating for each configuration (*Nat*). A Wilcoxon rank sum test shows that all systems outperform the FLM system returning the top hypothesis of the search models, with no reranking ($p < 0.0001$,

Table 6
Evaluation results for different reranking configurations. *Beam* = paraphrase selection beam (% of first best probability); *Mean n* = mean number of paraphrases per act; *Total n* = total number of paraphrases used for evaluation; *Nat* = mean naturalness rating over the generated paraphrase set. The last 3 columns indicate the significance of the difference in naturalness according to a two-tailed Wilcoxon rank sum test (*$p < 0.05$, **$p < 0.01$, ***$p < 0.001$).

Reranking method	Beam	Mean *n*	Total *n*	Nat	p_{base}	p_{FLM_0}	p_{FLM_8}
No reranking (base)	0%	1.05	723	3.16	n/a	***	***
FLM_0	0%	1.08	744	**3.83**	***	n/a	
FLM_8	8%	1.59	1,097	**3.78**	***		n/a
FLM_{15}	15%	2.12	1,465	3.67	***	***	*
FLM_{15} global	15%	2.03	1,405	3.68	***	**	*
Avg perceptron	n/a	1.46	1,012	3.68	***	**	*
Last perceptron	n/a	1.91	1,317	3.53	***	***	***

two-tailed).[5] We find that the best performance is obtained using the FLM reranking models, with an average naturalness of 3.83 when only considering the top hypothesis (FLM_0), compared with 3.16 without any reranking (*base*). Whereas the automated evaluation in Section 5.1 predicted an optimal selection beam of 8%, we find that the average naturalness decreases to 3.78 when taking the average over all paraphrases within that beam; however, the decrease in naturalness is not significant over 1,097 samples ($p = 0.33$). Because these results do not take the *coverage* of the generated paraphrase set into account, such a nonsignificant decrease in naturalness is encouraging, as it suggests that the naturalness of the paraphrases produced are close to the first-best. Using a larger selection beam of 15% increases coverage further but produces a significantly lower naturalness than both the FLM_0 and FLM_8 systems ($p < 0.01$ and $p < 0.05$, respectively). While we expected that sharing realization models across dialogue act types would help generalize, overall we find that using one realization model per dialogue act type does not perform significantly worse than global realization models (FLM_{15} global), although the former greatly reduces the number of model parameters.

Results show that the perceptron rerankers significantly improve naturalness over the no reranking baseline ($p < 0.0001$). We find that using the averaged weight vector produces a smaller set of paraphrases that are perceived as more natural ($p < 0.01$), confirming the improvement previously observed for the part-of-speech tagging task (Collins 2002a). However, results show that both the FLM_0 and FLM_8 systems outperform the perceptron-based systems ($p < 0.01$ and $p < 0.05$, respectively), and the FLM_8 system produces slightly more paraphrases. We find that the averaged perceptron reranking model produces utterances that are comparable to an FLM selection beam of 15%; although for the same level of naturalness, the thresholded FLM produces 2.03 utterances on average, as opposed to 1.46 for the perceptron.

Overall, this first human evaluation suggests that the FLM reranker with an 8% selection beam offers the best trade-off between utterance naturalness and paraphrasal variation.

7.3 Human Evaluation from Dialogue Extracts

Although a text-based evaluation gives a good insight into the level of naturalness of a generated paraphrase set, it does not evaluate whether differences in naturalness can be perceived in a spoken dialogue context, nor does it evaluate the effect of the linguistic variation resulting from the use of multiple paraphrases within a dialogue. In this regard, this section evaluates the following three hypotheses: (a) the learned generators can produce language perceived as natural in a dialogue context; (b) varying the paraphrases used throughout the dialogue improves the system's naturalness; and (c) this increase in naturalness makes the user more willing to interact with the system.

We test these hypotheses by conducting a series of *observer-based* listening tests comparing dialogue extracts in which the system utterances have been regenerated and resynthesized. The original dialogues were collected over the phone during a task-based evaluation of the Hidden Information State dialogue manager (Young et al. 2010) on the CamInfo domain, using a handcrafted rule-based language generator. Each utterance is

5 Note that a Wilcoxon signed rank paired test cannot be used because each system can produce a different number of utterances. As a result the reported significance is an approximation, since the samples may include examples generated from the same input.

synthesized using an HMM-based text-to-speech engine trained on the AWB voice of the ARCTIC data set using the HTS toolkit (Tokuda et al. 2000).

Our evaluation first compares different generation methods in a pairwise fashion: (a) the FLM reranking method with *n*-best outputs sampled from an 8% selection beam (*FLM n-best*); (b) the averaged kernelized perceptron reranking method with uniform sampling over positive predictions (*Perceptron*); and (c) the single output of the handcrafted rule-based generator (*Handcrafted*). The handcrafted generator is an extension of the SPaRKy sentence planner (Stent, Prasad, and Walker 2004), which associates each dialogue act with a content plan tree combining syntactic templates with rhetorical structure relations. The syntactic templates are aggregated two-by-two in a bottom–up fashion by trying different clause-combining operations (e.g., by inserting a conjunction, merging identical subjects, or associating each template with distinct sentences). The aggregated syntactic tree is then converted into a flat string using the RealPro surface realizer (Lavoie and Rambow 1997). The handcrafted generator has been tuned over several months to produce natural utterances for all possible input acts; we therefore treat it as a gold standard in our evaluation.

We also compare the FLM reranking approach with *n*-best outputs with an identical system that always selects the top realization at each turn (*FLM first-best*). In order to maximize the effect of generated linguistic variation, we do not sample paraphrases that were already chosen during the previous dialogue turns, unless there are no remaining paraphrases for that dialogue act. A total of 255 dialogues were regenerated for each system. In order to facilitate the listener's task while maintaining some aspect of the dialogue context, the dialogues were split into chunks consisting of the two consecutive system turns, concatenated with the corresponding prerecorded user turn. In order to make the dialogue extracts more intelligible, regenerated system turns are concatenated with the user turns with no speech overlap.

For each system pair, 600 dialogue extracts were randomly selected for evaluation out of all the regenerated dialogues. The raters are presented with four pairs of dialogue extracts at a time, which only differ by their system prompts. For each dialogue pair, they are asked to listen to both sound clips and evaluate (a) which system is the most natural (*naturalness* score), and (b) which system they would rather interact with (*user preference* score), as illustrated in Figure 16. Participants were native speakers of English recruited through Amazon Mechanical Turk, and geographically restricted to the USA. Although the British TTS voice used might affect overall perceptions of naturalness of U.S. judges, it should not introduce any bias within the system comparison as the same voice was used for each system. Each dialogue extract was rated by a single participant, and each participant could rate between 4 and 100 dialogue extract pairs. As a result, between 55 and 64 participants took part in the evaluation of each system pair.

7.3.1 Results. Table 7 summarizes the results of the preference tests. The naturalness and user preference scores represent the percentage of times the judges selected a given system over the other. A binomial test suggests that the judges did not prefer

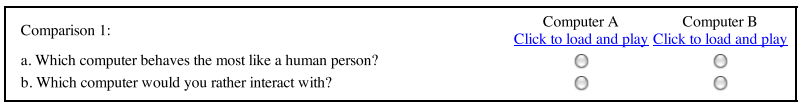


Figure 16
Human evaluation interface for comparing resynthesized dialogue extracts. Each crowdsourced evaluation task consisted of four pairwise system comparisons.

Table 7
Naturalness and user preference percentage of the best performing systems for each pairwise system comparison (winners in **bold**). Significance was computed using a two-tailed binomial test (*$p < 0.05$, **$p < 0.01$, ***$p < 0.001$).

System A	System B	Nat	Pref
FLM *n*-best	**Handcrafted**	52.2	52.3
FLM *n*-best	FLM 1-best	**57.3*** **	**54.2***
FLM *n*-best	Perceptron	51.2	52.7
Perceptron	**Handcrafted**	**54.2***	**54.2***

the handcrafted gold over the FLM reranker with *n*-best outputs, as no significance was reached over 600 comparisons ($p < 0.05$). However, the judges preferred the handcrafted generator over the perceptron reranker, possibly because it was also perceived as significantly more natural ($p < 0.05$). No significance was found when comparing the FLM reranker with the perceptron reranker, although most judges preferred the former, hence confirming results from the text-based evaluation. Finally, the FLM reranker with *n*-best outputs was perceived as significantly more natural than the same system with first-best output only ($p < 0.001$). Furthermore, results confirm that the judges would rather interact with the *n*-best system ($p < 0.05$). This result is interesting, as the *n*-best generation approach has a higher risk of selecting ungrammatical outputs compared with the first-best approach. However, our results show that despite that risk, judges prefer the *n*-best system, which suggests that data-driven paraphrase generation is beneficial in dialogue.

It is important to note that crowdsourced evaluations can lead to additional noise compared with standard lab-based evaluation, mostly due to the possibility of uncooperative evaluators. However, the randomization of the order of the evaluated utterances ensures that such noise does not bias the results towards one system. It is therefore likely that a more controlled evaluation would have revealed even more significant results.

8. Discussion and Conclusion

This article presents and evaluates BAGEL, a statistical language generator that can be trained entirely from data, with no handcrafting required beyond the semantic annotation. All the required subtasks (i.e., content ordering, aggregation, lexical selection, and realization) are performed implicitly through a search over Factored Language Models. We propose a stack-based semantic representation at the phrase level, which is expressive enough to generate natural utterances from *unseen* inputs, yet simple enough for data to be collected from a large set of untrained annotators with minimal manual correction and normalization. Results show that this approach outperforms utterance class LM methods on our data.

In order to make the Viterbi decoding tractable for real world dialogue applications, we limit the context-size of the decoding FLMs and rerank their *n*-best output using large-context reranking models. We investigate two types of reranking models, namely, (a) generatively trained FLM rerankers and (b) discriminatively trained structured perceptron models. The perceptron learns a discriminant function weighting local feature counts over the full utterance. By kernelizing the perceptron algorithm, the discriminant function is implicitly made dependent on a larger set of feature combinations (e.g., a polynomial kernel contains the products of each FLM context feature). Although our

results show that the perceptron reranking step is a viable alternative, we find that the large context FLM generalizes better on unseen data. This could be a consequence of the fact that some of the training examples of our algorithm are generated from the same input, and non-independently distributed data is likely to affect generalization error. A possible solution to this issue is to only allow a single weight update per input by moving the weight vector closer to the features of the lowest ranked reference paraphrase, and away from the highest ranked non-reference utterance. However, selecting the final paraphrase set would require a cut-off threshold that would need to be learned separately. Future work should also investigate the use of the large margin criterion instead of the perceptron criterion, which is more costly to compute but less likely to overfit; and it can minimize arbitrary loss functions (Tsochantaridis et al. 2004). Finally, the decoding models could also be learned discriminatively, for example, by learning to predict phrase sequences using maximum entropy Markov models or conditional random fields.

It is important to note that the *n*-best system evaluated in Section 7.3 is biased against exact repetitions (i.e., verbatim repetitions are prohibited unless all paraphrases have been generated). Our *n*-best system does not model the case in which verbatim repetitions could be used as an emphasis device. This could be addressed by adding a semantic element specifying that a specific phrase should be repeated for emphasis purposes. Because the *n*-best system did not implement that functionality, we believe that the preference for the *n*-best system could be increased when modeling exact repetitions. Apart from the case of emphasis, we believe that paraphrasing is generally more natural in dialogue contexts. Although this claim is difficult to evaluate, Torrance, Lee, and Olson (1992) have shown that children under 6 fail to distinguish between verbatim repetitions and paraphrases (i.e., before they learn to read). This result suggests that there might not be any additional cognitive load from using paraphrases in dialogue.

An important aspect of this work is that BAGEL's coarse-grained semantics allowed us to use crowdsourcing to collect semantically annotated utterances from untrained annotators. A first implication is that such methods could dramatically reduce development time of NLG systems, while improving scalability to large domains. Future work should therefore evaluate whether the same performance can be achieved in other task-oriented domains. Furthermore, although this work treats the training set as fixed, recent work has shown that active learning can further improve the efficiency of the data collection process (Mairesse et al. 2010).

We believe that the granularity of our semantic representation—which is defined by the attributes of the entity of interest in our domain—is expressive enough for a large range of information presentation systems. Although it is not as expressive as first-order logic, BAGEL implements the *all*, *none*, and *only* quantifiers by treating the quantifier as any other stack concept (see Figure 2 and rows 4 and 5 in Table 4). A limitation is that currently BAGEL can only present entities satisfying the same set of constraints within a dialogue act, for example, *X and Y are French restaurants near King's College*. Future work should focus on extending our semantic representation to include contrastive or justificative statements by allowing the presentation of entity-dependent attributes; for example, *X is near King's College however Y is close to the train station* or *You might be interested in X because it is cheap and near the VUE cinema*. Previous work in NLG has represented such statements using discourse relations from Mann and Thompson's Rhetorical Structure Theory (1988) as part of the sentence planning process (Walker, Rambow, and Rogati 2002; Stent, Prasad, and Walker 2004; Stent and Molina 2009). Hence BAGEL's expressiveness could be improved by including discourse relations as

part of the semantic tree and corresponding stack sequence. For example, *Charlie Chan is located near the Regal however Jinling is near King's College* could be represented by including a CONTRAST discourse relation to produce the tree in Figure 17. As this would require increasing the stack depth, experiments with new back-off strategies are likely to be required to confirm that BAGEL can generalize to support such discourse relations. Although adopting a formalism such as RST would increase BAGEL's expressiveness, it is also important to note that it would also raise the level of expertise required for annotating training utterances, thus potentially making it more difficult to rely on crowdsourcing for collecting training examples. There is thus a trade-off between the complexity of the semantic annotation and the amount of annotated data that can be realistically collected. While we believe the granularity of our semantic scheme offers a good balance for dialogue system applications, more research is needed to establish whether more fine-grained semantics can yield a sufficient amount of data in arbitrary domains.

The generation of utterances from arbitrary semantic symbols can be difficult for annotators. The BAGEL framework requires the ontology to be designed such that it can be easily annotated. Note that the same requirement exists for collecting data for the system's natural language understanding component. Annotation errors can typically be smoothed out by the statistical model; however, systematic errors due to ambiguities in the annotation schema can affect system performance. A consequence is that the annotation schema might require multiple iterations, based on the observed performance. We believe that most misunderstandings can be resolved by renaming semantic concepts, or by presenting example utterances to the annotators.

Crowdsourcing our data collection from a large range of annotators also provides us with a varied set of training paraphrases. Even without explicitly collecting multiple utterances for a single dialogue act, identical semantic concepts are typically associated with different realization phrases across dialogue acts. We believe that statistical NLG methods have the potential to learn to reproduce that variability at no extra cost. A further contribution of this article is therefore to present and evaluate two methods for learning to generate paraphrases in dialogue: (a) by thresholding the *n*-best output of FLM reranking models and (b) by using a perceptron reranker to learn a decision boundary between negative and positive utterances in the training set. Whereas NLG components are typically evaluated from text outputs only, we evaluate both

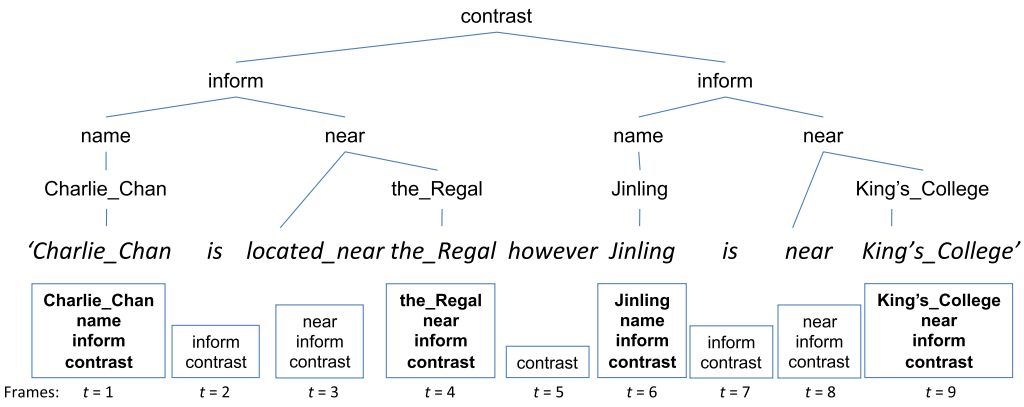


Figure 17
Example utterance and semantic alignment including a contrastive discourse relation. Mandatory stacks are in **bold**.

paraphrase generation methods within the context of dialogue system interaction. A first result is that human judges do not perceive the resynthesized outputs as significantly less natural than the outputs of a highly tuned handcrafted gold standard. This result confirms that BAGEL can successfully learn to generate utterances over a large, real-world domain. Furthermore, we find that a system varying its output by sampling from a thresholded *n*-best list is perceived more favorably than a system always returning the first-best utterance. These results need to be confirmed by a task-based dialogue system evaluation; but they suggest that users prefer systems producing varied linguistic outputs, which is contrary to the intuition that users are more comfortable with machines conversing in a predictable, repetitive, machine-like way.

Acknowledgments

This research was partly funded by the EU FP7 Programme under grant agreement 216594 (CLASSiC project: `www.classic-project.org`).

References

Angeli, Gabor, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of EMNLP*, pages 502–512, Cambridge, MA.

Bangalore, Srinivas and Owen Rambow. 2000. Exploiting a probabilistic hierarchical model for generation. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING)*, pages 42–48, Saarbrücken.

Bannard, Colin and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 597–604, Ann Arbor, MI.

Barzilay, Regina and Kathleen McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 50–57, Toulouse.

Belz, Anja. 2008. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Natural Language Engineering*, 14(4):431–455.

Bilmes, Jeff and Katrin Kirchhoff. 2003. Factored language models and generalized parallel backoff. In *Proceedings of HLT-NAACL, Short Papers*, pages 4–6, Edmonton.

Bulyko, Ivan and Mari Ostendorf. 2002. Efficient integrated response generation from multiple targets using weighted finite state transducers. *Computer Speech and Language*, 16(3-4):533–550.

Cahill, Aoife and Josef van Genabith. 2006. Robust PCFG-based generation using automatically acquired LFG approximations. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1,033–1,044, Sydney.

Chambers, Nathanael and James Allen. 2004. Stochastic language generation in a dialogue system: Toward a domain independent generator. In *Proceedings 5th SIGdial Workshop on Discourse and Dialogue*, pages 9–18, Cambridge, MA.

Collins, Michael. 2002a. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithm. In *Proceedings of EMNLP*, pages 1–8, Philadelphia, PA.

Collins, Michael. 2002b. Ranking algorithms for named-entity extraction: Boosting and the voted perceptron. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 489–496, Philadelphia, PA.

Collins, Michael and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 111–118, Barcelona.

Espinosa, Dominic, Michael White, and Dennis Mehay. 2008. Hypertagging: Supertagging for surface realization with CCG. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 183–191, Columbus, OH.

Foster, Mary Ellen and Michael White. 2005. Assessing the impact of adaptive generation in the COMIC multimodal dialogue system. In *Proceedings of the IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, pages 24–31, Edinburgh.

He, Yulan and Steve Young. 2005. Semantic processing using the Hidden Vector State

model. *Computer Speech & Language*, 19(1):85–106.
Isard, Amy, Carsten Brockmann, and Jon Oberlander. 2006. Individuality and alignment in generated dialogues. In *Proceedings of INLG*, pages 22–29, Sydney.
Kondadadi, Ravi, Blake Howald, and Frank Schilder. 2013. A statistical NLG framework for aggregated planning and realization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1,406–1,415, Sofia.
Konstas, Ioannis and Mirella Lapata. 2012. Concept-to-text generation via discriminative reranking. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 369–378, Jeju Island.
Langkilde, Irene and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 704–710, Montreal.
Langkilde-Geary, Irene. 2002. An empirical verification of coverage and correctness for a general-purpose sentence generator. In *Proceedings of the International Conference on Natural Language Generation*, pages 17–24, Harriman, NY.
Lavoie, Benoit and Owen Rambow. 1997. A fast and portable realizer for text generation systems. In *Proceedings of the 3rd Conference on Applied Natural Language Processing*, pages 265–268, Washington, DC.
Lefèvre, Fabrice. 2006. A DBN-based multi-level stochastic spoken language understanding system. In *Proceedings of SLT*, pages 78–81, Palm Beach, Aruba.
Lin, Dekang and Patrick Pantel. 2001. DIRT—discovery of inference rules from text. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 323–328, San Francisco, CA.
Lu, Wei, Hwee Tou Ng, and Wee Sun Lee. 2009. Natural language generation with tree conditional random fields. In *Proceedings of EMNLP*, pages 400–409, Edinburgh.
Mairesse, François, Milica Gašić, Filip Jurčíček, Simon Keizer, Blaise Thomson, Kai Yu, and Steve Young. 2010. Phrase-based statistical language generation using graphical models and active learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1,552–1,561, Uppsala.
Mairesse, François and Marilyn A. Walker. 2008. Trainable generation of Big-Five personality styles through data-driven parameter estimation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 165–173, Columbus, OH.
Mairesse, François and Marilyn A. Walker. 2011. Controlling user perceptions of linguistic style: Trainable generation of personality traits. *Computational Linguistics*, 37(3):455–488.
Mann, William C. and Sandra A. Thompson. 1988. Rhetorical structure theory. Toward a functional theory of text organization. *Text*, 8(3):243–281.
Nakanishi, Hiroko, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic methods for disambiguation of an HPSG-based chart generator. In *Proceedings of the 9th International Workshop on Parsing Technology*, pages 93–102, Vancouver.
Nakatsu, Crystal and Michael White. 2006. Learning to say it well: Reranking realizations by predicted synthesis quality. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1,113–1,120, Sydney.
Och, Franz Josef and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.
Oh, Alice H. and Alexander I. Rudnicky. 2002. Stochastic natural language generation for spoken dialog systems. *Computer Speech and Language*, 16:387–407.
Paiva, Daniel S. and Roger Evans. 2005. Empirically-based control of natural language generation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 58–65, Ann Arbor, MI.
Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318, Philadelphia, PA.
Pon-Barry, Heather, Karl Schultz, Elizabeth Owen Bratt, Brady Clark, and Stanley Peters. 2006. Responding to student uncertainty in spoken tutorial dialogue systems. *International Journal of Artificial Intelligence in Education*, 16:171–194.
Rabiner, Lawrence R. 1989. Tutorial on Hidden Markov Models and selected

applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285.
Ratnaparkhi, Adwait. 2002. Trainable approaches to surface natural language generation and their application to conversational dialog systems. *Computer Speech and Language*, 16(3-4):435–455.
Reiter, Ehud and Anja Belz. 2009. An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Computational Linguistics*, 25:529–558.
Rieser, Verena and Oliver Lemon, 2010. Natural language generation as planning under uncertainty for spoken dialogue systems. In E. Krahmer & M. Theune, editors, *Empirical Methods in Natural Language Generation*, Springer, Heidelberg, pages 105–120.
Schatzmann, Jost, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 2007. Agenda-based user simulation for bootstrapping a POMDP dialogue system. In *Proceedings of HLT-NAACL, Short Papers*, pages 149–152, Rochester, NY.
Stent, Amanda and Martin Molina. 2009. Evaluating automatic extraction of rules for sentence plan construction. In *Proceedings of the SIGdial Conference on Discourse and Dialogue*, pages 290–297, London.
Stent, Amanda, Rashmi Prasad, and Marilyn A. Walker. 2004. Trainable sentence planning for complex information presentation in spoken dialog systems. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 79–86, Barcelona.
Stone, Matthew, 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. In *Proceedings of SIGGRAPH*, pages 506–513, Los Angeles, CA.
Thomson, Blaise and Steve Young. 2010. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech & Language*, 24(4):562–588.
Thomson, Blaise, Kai Yu, Simon Keizer, Milica Gašić, Filip Jurčíček, François Mairesse, and Steve Young. 2010. Bayesian dialogue system for the Let's Go spoken dialogue challenge. In *Proceedings of SLT, Special Session: The Spoken Dialogue Challenge*, pages 460–465, Berkeley, CA.
Tokuda, Keiichi, Takayoshi Yoshimura, Takashi Masuko, Takao Kobayashi, and Tadashi Kitamura. 2000. Speech parameter generation algorithms for HMM-based speech synthesis. In *Proceedings of ICASSP*, pages 1,315–1,318, Istanbul.
Torrance, Nancy, Elizabeth Lee, and David R. Olson. 1992. The development of the distinction between paraphrase and exact wording in the recognition of utterances. In *Proceedings of the Annual Meeting of the American Educational Research Association*, pages 1–11, San Francisco, CA.
Tsochantaridis, Ioannis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support vector learning for interdependent and structured output spaces. In *Proceedings of ICML*, pages 104–112, Bauff.
Varges, Sebastian and Chris Mellish. 2001. Instance-based natural language generation. In *Proceedings of the Annual Meeting of the North American Chapter of the ACL (NAACL)*, pages 1–8, Pittsburgh, PA.
Walker, Marilyn A., Owen Rambow, and Monica Rogati. 2002. Training a sentence planner for spoken dialogue using boosting. *Computer Speech and Language*, 16(3-4):409–433.
White, Michael, Rajakrishnan Rajkumar, and Scott Martin. 2007. Towards broad coverage surface realization with CCG. In *Proceedings of the Workshop on Using Corpora for NLG: Language Generation and Machine Translation*, pages 22–30, Copenhagen.
Wong, Yuk Wah and Raymond J. Mooney. 2007. Generation by inverting a semantic parser that uses statistical machine translation. In *Proceedings of HLT-NAACL*, pages 172–179, Rochester, NY.
Young, Steve, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. 2010. The Hidden Information State model: A practical framework for POMDP-based spoken dialogue management. *Computer Speech and Language*, 24(2):150–174.