Unsupervised Concept-to-text Generation with Hypergraphs

Ioannis Konstas and **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
i.konstas@sms.ed.ac.uk, mlap@inf.ed.ac.uk

Abstract

Concept-to-text generation refers to the task of automatically producing textual output from non-linguistic input. We present a joint model that captures content selection ("what to say") and surface realization ("how to say") in an unsupervised domain-independent fashion. Rather than breaking up the generation process into a sequence of local decisions, we define a probabilistic context-free grammar that globally describes the inherent structure of the input (a corpus of database records and text describing some of them). We represent our grammar compactly as a weighted hypergraph and recast generation as the task of finding the best derivation tree for a given input. Experimental evaluation on several domains achieves competitive results with state-of-the-art systems that use domain specific constraints, explicit feature engineering or labeled data.

1 Introduction

Concept-to-text generation broadly refers to the task of automatically producing textual output from non-linguistic input (Reiter and Dale, 2000). Depending on the application and the domain at hand, the input may assume various representations including databases of records, expert system knowledge bases, simulations of physical systems and so on. Figure 1 shows input examples and their corresponding text for three domains, air travel, sportscasting and weather forecast generation.

A typical concept-to-text generation system implements a pipeline architecture consisting of three core stages, namely text planning (determining the content and structure of the target text), sentence planning (determining the structure and lexical content of individual sentences), and surface realization (rendering the specification chosen by the sentence planner into a surface string). Traditionally, these components are hand-engineered in order to generate high quality text, however at the expense of portability and scalability. It is thus no surprise that recent years have witnessed a growing interest in automatic methods for creating trainable generation components. Examples include learning which database records should be present in a text (Duboue and McKeown, 2002; Barzilay and Lapata, 2005) and how these should be verbalized (Liang et al., 2009). Besides concentrating on isolated components, a few approaches have emerged that tackle concept-to-text generation end-to-end. Due to the complexity of the task, most models simplify the generation process, e.g., by creating output that consists of a few sentences, thus obviating the need for document planning, or by treating sentence planning and surface realization as one component. A common modeling strategy is to break up the generation process into a sequence of local decisions, each learned separately (Reiter et al., 2005; Belz, 2008; Chen and Mooney, 2008; Angeli et al., 2010; Kim and Mooney, 2010).

In this paper we describe an end-to-end generation model that performs content selection and surface realization jointly. Given a corpus of database records and textual descriptions (for some of them), we define a probabilistic context-free grammar (PCFG) that captures the structure of the database and how it can be rendered into natural

2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 752–761,
Montréal, Canada, June 3-8, 2012.

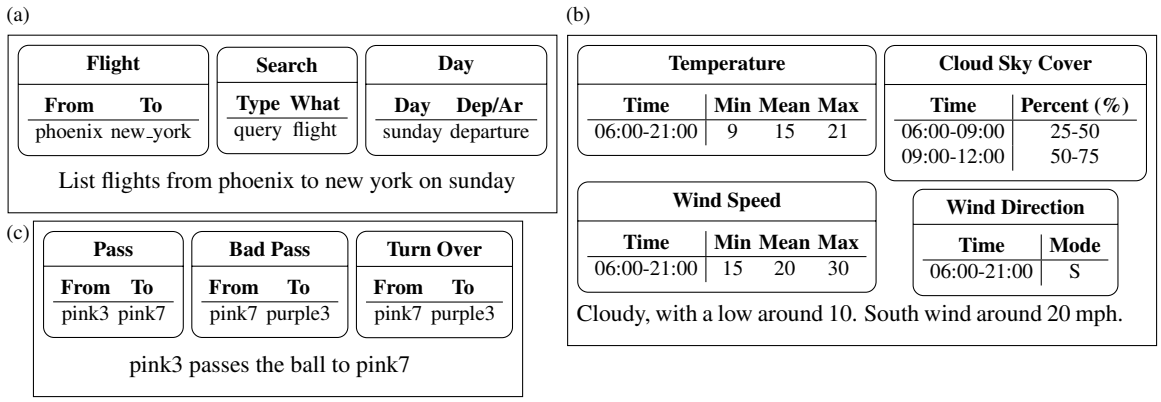

Figure 1: Input-output examples for (a) query generation in the air travel domain, (b) weather forecast generation, and (c) sportscasting.

language. This grammar represents a set of trees which we encode compactly using a weighted hypergraph (or packed forest), a data structure that defines a probability (or weight) for each tree. Generation then boils down to finding the best derivation tree in the hypergraph which can be done efficiently using the Viterbi algorithm. In order to ensure that our generation output is fluent, we intersect our grammar with a language model and perform decoding using a dynamic programming algorithm (Huang and Chiang, 2007).

Our model is conceptually simpler than previous approaches and encodes information about the domain and its structure globally, by considering the input space *simultaneously* during generation. Our only assumption is that the input must be a set of records essentially corresponding to database-like tables whose columns describe fields of a certain type. Experimental evaluation on three domains obtains results competitive to the state of the art without using any domain specific constraints, explicit feature engineering or labeled data.

2 Related Work

Our work is situated within the broader class of data-driven approaches to content selection and surface realization. Barzilay and Lapata (2005) focus on the former problem which they view as an instance of collective classification (Barzilay and Lapata, 2005). Given a corpus of database records and texts describing some of them, they learn a content selection model that simultaneously optimizes local label assignments and their pairwise relations. Building on this work, Liang et al. (2009) present a hierarchical hidden semi-Markov generative model that first determines which facts to discuss and then generates words from the predicates and arguments of the chosen facts.

A few approaches have emerged more recently that combine content selection and surface realization. Kim and Mooney (2010) adopt a two-stage approach: using a generative model similar to Liang et al. (2009), they first decide what to say and then verbalize the selected input with WASP^{-1}, an existing generation system (Wong and Mooney, 2007). In contrast, Angeli et al. (2010) propose a unified content selection and surface realization model which also operates over the alignment output produced by Liang et al. (2009). Their model decomposes into a sequence of discriminative local decisions. They first determine which records in the database to talk about, then which fields of those records to mention, and finally which words to use to describe the chosen fields. Each of these decisions is implemented as a log-linear model with features learned from training data. Their surface realization component is based on templates that are automatically extracted and smoothed with domain-specific constraints in order to guarantee fluent output. Other related work (Wong and Mooney, 2007; Lu and Ng, 2011). has focused on generating natural language sentences from logical form (i.e., lambda-expressions) using mostly synchronous context-free grammars (SCFGs).

Similar to Angeli et al. (2010), we also present an end-to-end system that performs content selection and surface realization. However, rather than breaking up the generation task into a sequence of local decisions, we optimize what to say and how to say simultaneously. We do not learn mappings from a logical form, but rather focus on input which is less constrained, possibly more noisy and with a looser structure. Our key insight is to convert the set of database records serving as input to our generator into a PCFG that is neither hand crafted nor domain specific but simply describes the structure of the database. The approach is conceptually simple, does not rely on discriminative training or any feature engineering. We represent the grammar and its derivations compactly as a weighted hypergraph which we intersect with a language model in order to generate fluent output. This allows us to easily port surface generation to different domains without having to extract new templates or enforce domain specific constraints.

3 Problem Formulation

We assume our generator takes as input a set of database records $\mathbf{d}$ and produces text $\mathbf{w}$ that verbalizes some of these records. Each record $r \in \mathbf{d}$ has a type $r.t$ and a set of fields f associated with it. Fields have different values $f.v$ and types $f.t$ (i.e., integer or categorical). For example, in Figure 1b, wind speed is a record type with four fields: `time`, `min`, `mean`, and `max`. The values of these fields are `06:00-21:00`, `15`, `20`, and `30`, respectively; the type of `time` is categorical, whereas all other fields are integers.

During training, our algorithm is given a corpus consisting of several *scenarios*, i.e., database records paired with texts like those shown in Figure 1. In the weather forecast domain, a scenario corresponds to weather-related measurements of temperature, wind, speed, and so on collected for a specific day and time (e.g., day or night). In sportscasting, scenarios describe individual events in the soccer game (e.g., passing or kicking the ball). In the air travel domain, scenarios comprise of flight-related details (e.g., origin, destination, day, time). Our goal then is to reduce the tasks of content selection and surface realization into a common probabilistic parsing problem. We do this by abstracting the structure of the database (and accompanying texts) into a PCFG whose probabilities are learned from training data.[1] Specifically, we convert the database into rewrite rules and represent them as a weighted directed hypergraph (Gallo et al., 1993). Instead of learning the probabilities on the PCFG, we directly compute the weights on the hyperarcs using a dynamic program similar to the inside-outside algorithm (Li and Eisner, 2009). During testing, we are given a set of database records without the corresponding text. Using the trained grammar we compile a hypergraph specific to this test input and decode it approximately via cube pruning (Chiang, 2007).

The choice of the hypergraph framework is motivated by at least three reasons. Firstly, hypergraphs can be used to represent the search space of most parsers (Klein and Manning, 2001). Secondly, they are more efficient and faster than the common CYK parser-based representation for PCFGs by a factor of more than ten (Huang and Chiang, 2007). And thirdly, the hypergraph representation allows us to integrate an n-gram language model and perform decoding efficiently using k-best Viterbi search, optimizing what to say and how to say at the same time.

3.1 Grammar Definition

Our model captures the inherent structure of the database with a number of CFG rewrite rules, in a similar way to how Liang et al. (2009) define Markov chains in the different levels of their hierarchical model. These rules are purely syntactic (describing the intuitive relationship between records, records and fields, fields and corresponding words), and could apply to any database with similar structure irrespectively of the semantics of the domain.

Our grammar is defined in Table 1 (rules (1)–(9)). Rule weights are governed by an underlying multinomial distribution and are shown in square brackets. Non-terminal symbols are in capitals and de-

[1] An alternative would be to learn a SCFG between the database input and the accompanying text. However, this would involve considerable overhead in terms of alignment (as the database and the text do not together constitute a clean parallel corpus, but rather a noisy comparable corpus), as well as grammar training and decoding using state-of-the art SMT methods, which we manage to avoid with our simpler approach.

1. $\mathrm{S} \rightarrow \mathrm{R}(start)$	$[Pr = 1]$
2. $\mathrm{R}(r_i.t) \rightarrow \mathrm{FS}(r_j, start)\ \mathrm{R}(r_j.t)$	$[P(r_j.t \mid r_i.t) \cdot \lambda]$
3. $\mathrm{R}(r_i.t) \rightarrow \mathrm{FS}(r_j, start)$	$[P(r_j.t \mid r_i.t) \cdot \lambda]$
4. $\mathrm{FS}(r, r.f_i) \rightarrow \mathrm{F}(r, r.f_j)\ \mathrm{FS}(r, r.f_j)$	$[P(f_j \mid f_i)]$
5. $\mathrm{FS}(r, r.f_i) \rightarrow \mathrm{F}(r, r.f_j)$	$[P(f_j \mid f_i)]$
6. $\mathrm{F}(r, r.f) \rightarrow \mathrm{W}(r, r.f)\ \mathrm{F}(r, r.f)$	$[P(w \mid w_{-1}, r, r.f)]$
7. $\mathrm{F}(r, r.f) \rightarrow \mathrm{W}(r, r.f)$	$[P(w \mid w_{-1}, r, r.f)]$
8. $\mathrm{W}(r, r.f) \rightarrow \alpha$	$[P(\alpha \mid r, r.f, f.t, f.v)]$
9. $\mathrm{W}(r, r.f) \rightarrow \mathrm{g}(f.v)$	$[P(\mathrm{g}(f.v).mode \mid r, r.f, f.t = int)]$

Table 1: Grammar rules and their weights shown in square brackets.

note intermediate states; the terminal symbol α corresponds to all words seen in the training set, and $\mathrm{g}(f.v)$ is a function for generating integer numbers given the value of a field f. All non-terminals, save the start symbol S, have one or more features (shown in parentheses) that act as constraints, similar to number and gender agreement constraints in augmented syntactic rules.

Rule (1) denotes the expansion from the start symbol S to record R, which has the special 'start' record type (hence the notation $\mathrm{R}(start)$). Rule (2) defines a chain between two consecutive records, i.e., going from a source record r_i to a target r_j. Here, $\mathrm{FS}(r_j, r_j.f)$ represents the set of fields of the target r_j, following the source record $\mathrm{R}(r_i)$. For example, the rule $\mathrm{R}(skyCover_1.t) \rightarrow \mathrm{FS}(temperature_1, start)\mathrm{R}(temperature_1.t)$ can be interpreted as follows. Given that we have talked about $skyCover_1$, we will next talk about $temperature_1$ and thus emit its corresponding fields. $\mathrm{R}(temperature_1.t)$ is a non-terminal place-holder for the continuation of the chain of records, and *start* in FS is a special boundary field between consecutive records. The weight of this rule is the bigram probability of two records conditioned on their record type, multiplied with a normalization factor λ. We have also defined a *null* record type i.e., a record that has no fields and acts as a smoother for words that may not correspond to a particular record. Rule (3) is simply an escape rule, so that the parsing process (on the record level) can finish.

Rule (4) is the equivalent of rule (2) at the field level, i.e., it describes the chaining of two consecutive fields f_i and f_j. Non-terminal $\mathrm{F}(r, r.f)$ refers to field f of record r. For example, the rule $\mathrm{FS}(windSpeed_1, min) \rightarrow \mathrm{F}(windSpeed_1, max)\mathrm{FS}(windSpeed_1, max)$, specifies that we should talk about the field *max* of record $windSpeed_1$, after talking about the field *min*. Analogously to the record level, we have also included a special *null* field type for the emission of words that do not correspond to a specific record field. Rule (6) defines the expansion of field F to a sequence of (binarized) words W, with a weight equal to the bigram probability of the current word given the previous word, the current record, and field. This is an attempt at capturing contextual dependencies between words over and above to integrating a language model during decoding (see Section 3.3).

Rules (8) and (9) define the emission of words and integer numbers from W, given a field type and its value. Rule (8) emits a single word from the vocabulary of the training set. Its weight defines a multinomial distribution over all seen words, for every value of field f, given that the field type is categorical or the special *null* field. Rule (9) is identical but for fields whose type is integer. Function $\mathrm{g}(f.v)$ generates an integer number given the field value, using either of the following six ways (Liang et al., 2009): identical to the field value, rounding up or rounding down to a multiple of 5, rounding off to the closest multiple of 5 and finally adding or subtracting some unexplained noise.[2] The weight is a multinomial over the six generation function *modes*, given the record field f.

3.2 Hypergraph Construction

So far we have defined a probabilistic grammar that captures the structure of a database **d** with records and fields as intermediate non-terminals, and words **w** (from the associated text) as terminals. Using this grammar and the CYK parsing algorithm, we could obtain the top scoring derivation of records and fields for a given input (i.e., a sequence of

[2]The noise is modeled as a geometric distribution.

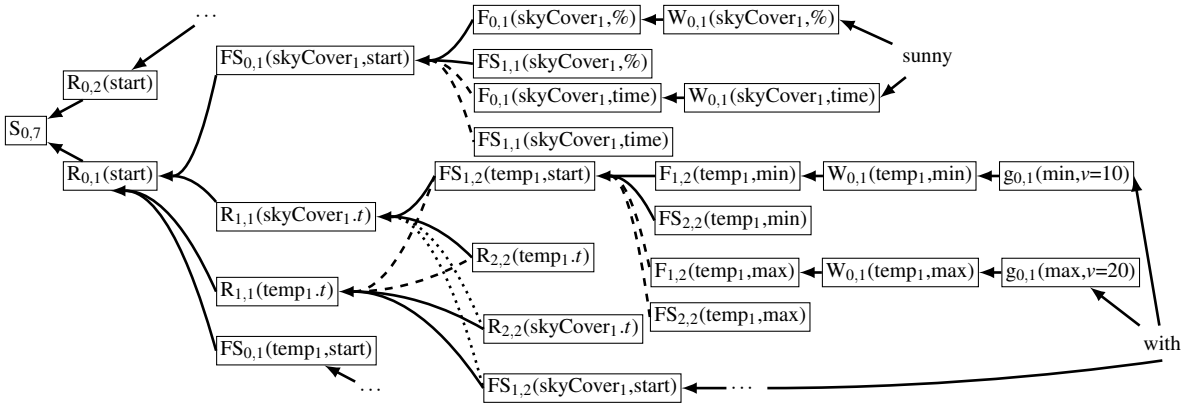

Figure 2: Partial hypergraph representation for the sentence *"Sunny with a low around 30 ."* For the sake of readability, we show a partial span on the first two words without weights on the hyperarcs.

words) as well as the optimal segmentation of the text, provided we have a trained set of weights. The inside-outside algorithm is commonly used for estimating the weights of a PCFG. However, we first transform the CYK parser and our grammar into a hypergraph and then compute the weights using inside-outside. Huang and Chiang (2005) define a weighted directed hypergraph as follows:

Definition 1 *An ordered hypergraph H is a tuple $\langle N, E, t, \boldsymbol{R} \rangle$, where N is a finite set of nodes, E is a finite set of hyperarcs and $\boldsymbol{R}$ is the set of weights. Each hyperarc $e \in E$ is a triple $e = \langle T(e), h(e), f(e) \rangle$, where $h(e) \in N$ is its head node, $T(e) \in N^*$ is a set of tail nodes and $f(e)$ is a monotonic weight function $\boldsymbol{R}_{|T(e)|}$ to $\boldsymbol{R}$ and $t \in N$ is a target node.*

Definition 2 *We impose the arity of a hyperarc to be $|e| = |T(e)| = 2$, in other words, each head node is connected with at most two tail nodes.*

Given a context-free grammar $G = \langle N, T, P, S \rangle$ (where N is the set of variables, T the set of terminals, P the set of production rules, and $S \in N$ the start symbol) and an input string $\mathbf{w}$, we can map the standard weighted CYK algorithm to a hypergraph as follows. Each node $[A, i, j]$ in the hypergraph corresponds to non-terminal A spanning words w_i to w_j of the input. Each rewrite rule $A \rightarrow BC$ in P, with three free indices $i < j < k$, is mapped to the hyperarc $\langle ((B, i, j), (C, j, k)), (A, i, k), f \rangle$, where $f = f((B, i, j)) f((C, j, k)) \cdot Pr(A \rightarrow BC)$.[3] The hypergraph can be thus viewed as a compiled lattice of the corresponding chart graph. Figure 2 shows an example hypergraph for a grammar defined on database input similar to Figure (1b).

In order to learn the weights on the hyperarcs we perform the following procedure iteratively in an EM fashion (Li and Eisner, 2009). For each training scenario we build its hypergraph representation. Next, we perform inference by calculating the inside and outside scores of the hypergraph, so as to compute the posterior distribution over its hyperarcs (E-step). Finally, we collectively update the posteriors on the parameters-weights, i.e., rule probabilities and emission multinomial distributions (M-step).

3.3 Decoding

In the framework outlined above, parsing an input string $\mathbf{w}$ (given some learned weights) boils down to traversing the hypergraph in a particular order. (Note that the hypergraph should be acyclic, which is always guaranteed by the grammar in Table 1). In generation, our aim is to verbalize an input scenario from a database $\mathbf{d}$ (see Figure 1). We thus find the best text by maximizing:

$$\arg\max_{w} P(\mathbf{w} \,|\, \mathbf{d}) = \arg\max_{w} P(\mathbf{w}) \cdot P(\mathbf{d} \,|\, \mathbf{w}) \qquad (1)$$

where $P(\mathbf{d} \,|\, \mathbf{w})$ is the decoding likelihood for a sequence of words $\mathbf{w}$, $P(\mathbf{w})$ is a measure of the quality of each output (given by a language model), and $P(\mathbf{w} \,|\, \mathbf{d})$ the posterior of the best output for database $\mathbf{d}$. Note that calculating $P(\mathbf{d} \,|\, \mathbf{w})$ requires deciding on the output length $|w|$. Rather than set-

[3] Similarly, rewrite rules of type $A \rightarrow B$ are mapped to the hyperarc $\langle (B, i, j), (A, i, j), f \rangle$, with $f = f((B, i, j)) \cdot Pr(A \rightarrow B)$.

ting w to a fixed length, we rely on a linear regression predictor that uses the counts of each record type per scenario as features and is able to produce variable length texts.

In order to perform decoding with an n-gram language model, we adopt Huang and Chiang's (2007) dynamic-programming algorithm for SCFG-based systems. Each node in the hypergraph is split into a set of compound items, namely *+LM items*. Each +LM item is of the form $(n^{a\star b})$, where a and b are boundary words of the generation string, and $\star$ is a place-holder symbol for an elided part of that string, indicating a sub-generation part ranging from a to b. An example +LM deduction of a single hyperarc of the hypergraph in Figure 2 using bigrams is:

$$\frac{\begin{array}{l}\mathrm{FS}_{1,2}(temp_1, start)^{low} : (w_1, g_1), \\ \mathrm{R}_{2,2}(temp_1.t)^{around\star degrees} : (w_2, g_2)\end{array}}{\mathrm{R}_{1,1}(skyCover_1.t)^{low\star degrees} : (w, g_1 g_2)} \quad (2)$$

$$w = w_1 + w_2 + e_w + P_{lm}(around \mid low) \quad (3)$$

where w_1, w_2 are node weights, g_1, g_2 are the corresponding sub-generations, e_w is the weight of the hyperarc and w the weight of the resulting +LM item. P_{lm} and $(n^{a\star b})$ are defined as in Chiang (2007) in a generic fashion, allowing extension to an arbitrary size of n-gram grammars.

Naive traversal of the hypergraph bottom-up would explore all possible +LM deductions along each hyperarc, and would increase decoding complexity to an infeasible $O(2^n n^2)$, assuming a trigram model and a constant number of emissions at the terminal nodes. To ensure tractability, we adopt cube pruning, a popular approach in syntax-inspired machine translation (Chiang, 2007). The idea is to use a beam-search over the intersection grammar coupled with the cube-pruning heuristic. The beam limits the number of derivations for each node, whereas cube-pruning further limits the number of +LM items considered for inclusion in the beam. Since $f(e)$ in Definition 1 is monotonic, we can select the k-best items without computing all possible +LM items.

Our decoder follows Huang and Chiang (2007) but importantly differs in the treatment of leaf nodes in the hypergraph (see rules (8) and (9)). In the SCFG context, the Viterbi algorithm consumes terminals from the source string in a bottom-up fashion and creates sub-translations according to the CFG rule that holds each time. In the concept-to-text generation context, however, we do not observe the words; instead, for each leaf node we emit the k-best words from the underlying multinomial distribution (see weights on rules (8) and (9)) and continue building our sub-generations bottom-up.

4 Experimental Design

Data We used our system to generate soccer commentaries, weather forecasts, and spontaneous utterances relevant to the air travel domain (examples are given in Figure 1). For the first domain we used the dataset of Chen and Mooney (2008), which consists of 1,539 scenarios from the 2001–2004 Robocup game finals. Each scenario contains on average $|\mathbf{d}| = 2.4$ records, each paired with a short sentence (5.7 words). This domain has a small vocabulary (214 words) and simple syntax (e.g., a transitive verb with its subject and object). Records in this dataset (henceforth ROBOCUP) were aligned manually to their corresponding sentences (Chen and Mooney, 2008). Given the relatively small size of this dataset, we performed cross-validation following previous work (Chen and Mooney, 2008; Angeli et al., 2010). We trained our system on three ROBOCUP games and tested on the fourth, averaging over the four train/test splits.

For weather forecast generation, we used the dataset of Liang et al. (2009), which consists of 29,528 weather scenarios for 3,753 major US cities (collected over four days). The vocabulary in this domain (henceforth WEATHERGOV) is comparable to ROBOCUP (345 words), however, the texts are longer ($|\mathbf{w}| = 29.3$) and more varied. On average, each forecast has 4 sentences and the content selection problem is more challenging; only 5.8 out of the 36 records per scenario are mentioned in the text which roughly corresponds to 1.4 records per sentence. We used 25,000 scenarios from WEATHERGOV for training, 1,000 scenarios for development and 3,528 scenarios for testing. This is the same partition used in Angeli et al. (2010).

For the air travel domain we used the ATIS dataset (Dahl et al., 1994), consisting of 5,426 scenarios. These are transcriptions of spontaneous utterances of users interacting with a hypothetical on-

	WEATHERGOV	ATIS	ROBOCUP
1-BEST	Near 57. Near 57. Near 57. Near 57. Near 57. Near 57. Near 57. Near 57. Near 57. Near 57. South wind.	What what what what flights from Denver Phoenix	Pink9 to to Pink7 kicks
k-BEST	As high as 23 mph. Chance of precipitation is 20. Breezy, with a chance of showers. Mostly cloudy, with a high near 57. South wind between 3 and 9 mph.	Show me the flights from Denver to Phoenix	Pink9 passes back to Pink7
ANGELI	A chance of rain or drizzle, with a high near 57. South wind between 3 and 9 mph.	Show me the flights leave from Nashville to Phoenix	Pink9 kicks to Pink7
HUMAN	A slight chance of showers. Mostly cloudy, with a high near 58. South wind between 3 and 9 mph, with gusts as high as 23 mph. Chance of precipitation is 20%.	List flights from Denver to Phoenix	Pink9 passes back to Pink7

Table 2: System output on WEATHERGOV, ATIS, and ROBOCUP (1-BEST, *k*-BEST, ANGELI) and corresponding human-authored text (HUMAN).

line flight booking system. We used the dataset introduced in Zettlemoyer and Collins (2007)[4] and automatically converted their lambda-calculus expressions to attribute-value pairs following the conventions adopted by Liang et al. (2009). For example, the scenario in Figure 1(a) was initially represented as: $\lambda x.flight(x) \wedge from(x, phoenix) \wedge to(x, new\_york) \wedge day(x, sunday)$.[5] In contrast to the two previous datasets, ATIS has a much richer vocabulary (927 words); each scenario corresponds to a single sentence (average length is 11.2 words) with 2.65 out of 19 record types mentioned on average. Following Zettlemoyer and Collins (2007), we trained on 4,962 scenarios and tested on ATIS NOV93 which contains 448 examples.

Model Parameters Our model has two parameters, namely the number of k grammar derivations considered by the decoder and the order of the language model. We tuned k experimentally on held-out data taken from WEATHERGOV, ROBOCUP, and ATIS, respectively. The optimal value was k=15 for WEATHERGOV, k=25 for ROBOCUP, and $k = 40$ for ATIS. For the ROBOCUP domain, we used a bigram language model which was considered sufficient given that the average text length is small. For WEATHERGOV and ATIS, we used a trigram language model.

System Comparison We evaluated two configurations of our system. A baseline that uses the top scoring derivation in each subgeneration (1-BEST) and another version which makes better use of our decoding algorithm and considers the best k derivations (i.e., 15 for WEATHERGOV, 40 for ATIS, and 25 for ROBOCUP). We compared our output to Angeli et al. (2010) whose approach is closest to ours and state-of-the-art on the WEATHERGOV domain. For ROBOCUP, we also compare against the best-published results (Kim and Mooney, 2010).

Evaluation We evaluated system output automatically, using the BLEU modified precision score (Papineni et al., 2002) with the human-written text as reference. In addition, we evaluated the generated text by eliciting human judgments. Participants were presented with a scenario and its corresponding verbalization and were asked to rate the latter along two dimensions: fluency (is the text grammatical and overall understandable?) and semantic correctness (does the meaning conveyed by the text correspond to the database input?). The subjects used a five point rating scale where a high number indicates better performance. We randomly selected 12 doc-

[4]The original corpus contains user utterances of single dialogue turns which would result in trivial scenarios. Zettlemoyer and Collins (2007) concatenate all user utterances referring to the same dialogue act, (e.g., book a flight), thus yielding more complex scenarios with longer sentences.

[5]The resulting dataset and a technical report describing the mapping procedure in detail are available from `http://homepages.inf.ed.ac.uk/s0793019/index.php?page=resources`

System	ROBOCUP BLEU	WEATHERGOV BLEU	ATIS BLEU
1-BEST	10.79	8.64	11.85
k-BEST	30.90	33.70	29.30
ANGELI	28.70	38.40	26.77
KIM-MOONEY	47.27	—	—

Table 3: BLEU scores on ROBOCUP (fixed content selection), WEATHERGOV, and ATIS.

System	ROBOCUP F	ROBOCUP SC	WEATHERGOV F	WEATHERGOV SC	ATIS F	ATIS SC
1-BEST	$2.47^{*\dagger}$	$2.33^{*\dagger}$	$1.82^{*\dagger}$	$2.05^{*\dagger}$	$2.40^{*\dagger}$	$2.46^{*\dagger}$
k-BEST	4.31^{*}	3.96^{*}	3.92^{*}	3.30^{*}	4.01	3.87
ANGELI	$4.03^{*\dagger}$	$3.70^{*\dagger}$	4.26^{*}	3.60^{*}	$3.56^{*\dagger}$	$3.33^{*\dagger}$
HUMAN	$4.47^{\dagger}$	$4.37^{\dagger}$	$4.61^{\dagger}$	$4.03^{\dagger}$	4.10	4.01

Table 4: Mean ratings for fluency (F) and semantic correctness (SC) on system output elicited by humans on ROBOCUP, WEATHERGOV, and ATIS (*: sig. diff. from HUMAN; †: sig. diff. from *k*-BEST.)

uments from the test set (for each domain) and generated output with our models (1-BEST and *k*-BEST) and Angeli et al.'s (2010) model (see Figure 2 for examples of system output). We also included the original text (HUMAN) as gold standard. We thus obtained ratings for 48 (12×4) scenario-text pairs for each domain. The study was conducted over the Internet using WebExp (Keller et al., 2009) and was completed by 114 volunteers, all self reported native English speakers.

5 Results

We conducted two experiments on the ROBOCUP domain. We first assessed the performance of our generator (*k*-BEST) on joint content selection and surface realization and obtained a BLEU score of 24.88. In comparison, the baseline's (1-BEST) BLEU score was 8.01. In a second experiment we forced the generator to use the gold-standard records from the database. This was necessary in order to compare with previous work (Angeli et al., 2010; Kim and Mooney, 2010).[6] Our results are summarized in Table 3. Overall, our generator performs better than the baseline and Angeli et al. (2010). We observe a substantial increase in performance compared to the joint content selection and surface realization setting. This is expected as the generator is faced with an easier task and there is less scope for error. Our model does not outperform Kim and Mooney (2010), however, this is not entirely surprising as their model requires considerable more supervision (e.g., during parameter initialization) and includes a post-hoc re-ordering component.

[6] Angeli et al. (2010) and Kim and Mooney (2010) fix content selection both at the record and field level. We let our generator select the appropriate fields, since these are at most two per record type and this level of complexity can be easily tackled during decoding.

With regard to WEATHERGOV, our generator improves over the baseline but lags behind Angeli et al. (2010). Since our system emits words based on a language model rather than a template, it displays more freedom in word order and lexical choice, and is thus penalized by BLEU when creating output that is overly distinct from the reference. On ATIS, our model outperforms both the baseline and Angeli et al. This is the most challenging domain with regard to surface realization with a vocabulary larger than ROBOCUP and WEATHERGOV by factors of 2.7 and 4.3, respectively.

The results of our human evaluation study are shown in Table 3. We carried out an Analysis of Variance (ANOVA) to examine the effect of system type (1-BEST, *k*-BEST, ANGELI, and HUMAN) on the fluency and semantic correctness ratings. Means differences were compared using a post-hoc Tukey test. On ROBOCUP, our system (*k*-BEST) is significantly better than the baseline (1-BEST) and ANGELI both in terms of fluency and semantic correctness ($a < 0.05$). On WEATHERGOV, our generator performs comparably to ANGELI on fluency and semantic correctness (the differences in the means are not statistically significant); 1-BEST is significantly worse than 15-BEST and ANGELI ($a < 0.05$). On ATIS, *k*-BEST is significantly more fluent and semantically correct than 1-BEST and ANGELI ($a < 0.01$). There was no statistically significant difference between the output of our system and the original ATIS sentences.

In sum, we observe that taking the *k*-best derivations into account boosts performance (the 1-BEST system is consistently worse). Our model is on par with ANGELI on WEATHERGOV but performs better on ROBOCUP and ATIS when evaluated both auto-

matically and by humans. In general, a large part of our output resembles the human text, which demonstrates that our simple language model yields coherent sentences (without any template engineering), at least for the domains under consideration.

6 Conclusions

We have presented an end-to-end generation system that performs both content selection and surface realization. Central to our approach is the encoding of generation as a parsing problem. We reformulate the input (a set of database records and text describing some of them) as a PCFG and show how to find the best derivation using the hypergraph framework. Despite its simplicity, our model is able to obtain performance comparable to the state of the art. We argue that our approach is computationally efficient and viable in practical applications. Porting the system to a different domain is straightforward, assuming a database and corresponding (unaligned) text. As long as the database is compatible with the structure of the grammar in Table 1, we need only retrain to obtain the weights on the hyperarcs and a domain specific language model.

Our model takes into account the k-best derivations at decoding time, however inspection of these shows that it often fails to select the best one. In the future, we plan to remedy this by using forest reranking, a technique that approximately reranks a packed forest of exponentially many derivations (Huang, 2008). We would also like to scale our model to more challenging domains (e.g., product descriptions) and to enrich our generator with some notion of discourse planning. An interesting question is how to extend the PCFG-based approach advocated here so as to capture discourse-level document structure.

Acknowledgments We are grateful to Percy Liang and Gabor Angeli for providing us with their code and data. We would also like to thank Luke Zettlemoyer and Tom Kwiatkowski for sharing their ATIS dataset with us and Frank Keller for his feedback on an earlier version of this paper.

References

Gabor Angeli, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 502–512, Cambridge, MA.

Regina Barzilay and Mirella Lapata. 2005. Collective content selection for concept-to-text generation. In *Proceedings of Human Language Technology and Empirical Methods in Natural Language Processing*, pages 331–338, Vancouver, British Columbia.

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

David L. Chen and Raymond J. Mooney. 2008. Learning to sportscast: A test of grounded language acquisition. In *Proceedings of International Conference on Machine Learning*, pages 128–135, Helsinki, Finland.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the atis task: the atis-3 corpus. In *Proceedings of the Workshop on Human Language Technology*, pages 43–48, Plainsboro, NJ.

Pablo A. Duboue and Kathleen R. McKeown. 2002. Content planner construction via evolutionary algorithms and a corpus-based fitness function. In *Proceedings of International Natural Language Generation*, pages 89–96, Ramapo Mountains, NY.

Giorgio Gallo, Giustino Longo, Stefano Pallottino, and Sang Nguyen. 1993. Directed hypergraphs and applications. *Discrete Applied Mathematics*, 42:177–201.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of the 9th International Workshop on Parsing Technology*, pages 53–64, Vancouver, British Columbia.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 144–151, Prague, Czech Republic.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08: HLT*, pages 586–594, Columbus, Ohio.

Frank Keller, Subahshini Gunasekharan, Neil Mayo, and Martin Corley. 2009. Timing accuracy of Web experiments: A case study using the WebExp software package. *Behavior Research Methods*, 41(1):1–12.

Joohyun Kim and Raymond Mooney. 2010. Generative alignment and semantic parsing for learning from ambiguous supervision. In *Proceedings of the 23rd Conference on Computational Linguistics*, pages 543–551, Beijing, China.

Dan Klein and Christopher D. Manning. 2001. Parsing and hypergraphs. In *Proceedings of the 7th International Workshop on Parsing Technologies*, pages 123–134, Beijing, China.

Zhifei Li and Jason Eisner. 2009. First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 40–51, Suntec, Singapore.

Percy Liang, Michael Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *roceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 91–99, Suntec, Singapore.

Wei Lu and Hwee Tou Ng. 2011. A probabilistic forest-to-string model for language generation from typed lambda calculus expressions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1611–1622, Edinburgh, UK.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania.

Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*. Cambridge University Press, New York, NY.

Ehud Reiter, Somayajulu Sripada, Jim Hunter, and Ian Davy. 2005. Choosing words in computer-generated weather forecasts. *Artificial Intelligence*, 167:137–169.

Yuk Wah Wong and Raymond Mooney. 2007. Generation by inverting a semantic parser that uses statistical machine translation. In *Proceedings of the Human Language Technology and the Conference of the North American Chapter of the Association for Computational Linguistics*, pages 172–179, Rochester, NY.

Luke Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 678–687, Prague, Czech Republic.