A Simple Domain-Independent Probabilistic Approach to Generation

Gabor Angeli
UC Berkeley
Berkeley, CA 94720
gangeli@berkeley.edu

Percy Liang
UC Berkeley
Berkeley, CA 94720
pliang@cs.berkeley.edu

Dan Klein
UC Berkeley
Berkeley, CA 94720
klein@cs.berkeley.edu

Abstract

We present a simple, robust generation system which performs content selection and surface realization in a unified, domain-independent framework. In our approach, we break up the end-to-end generation process into a sequence of local decisions, arranged hierarchically and each trained discriminatively. We deployed our system in three different domains—Robocup sportscasting, technical weather forecasts, and common weather forecasts, obtaining results comparable to state-of-the-art domain-specific systems both in terms of BLEU scores and human evaluation.

1 Introduction

In this paper, we focus on the problem of generating descriptive text given a world state represented by a set of database records. While existing generation systems can be engineered to obtain good performance on particular domains (e.g., Dale et al. (2003), Green (2006), Turner et al. (2009), Reiter et al. (2005), *inter alia*), it is often difficult to adapt them across different domains. Furthermore, *content selection* (what to say: see Barzilay and Lee (2004), Foster and White (2004), *inter alia*) and *surface realization* (how to say it: see Ratnaparkhi (2002), Wong and Mooney (2007), Chen and Mooney (2008), Lu et al. (2009), etc.) are typically handled separately. Our goal is to build a simple, flexible system which is domain-independent and performs content selection and surface realization in a unified framework.

We operate in a setting in which we are only given examples consisting of (i) a set of database records (input) and (ii) example human-generated text describing some of those records (output). We use the model of Liang et al. (2009) to automatically induce the correspondences between words in the text and the actual database records mentioned.

We break up the full generation process into a sequence of local decisions, training a log-linear classifier for each type of decision. We use a simple but expressive set of domain-independent features, where each decision is allowed to depend on the entire history of previous decisions, as in the model of Ratnaparkhi (2002). These long-range contextual dependencies turn out to be critical for accurate generation.

More specifically, our model is defined in terms of three types of decisions. The first type chooses records from the database (macro content selection)—for example, wind speed, in the case of generating weather forecasts. The second type chooses a subset of fields from a record (micro content selection)—e.g., the minimum and maximum temperature. The third type chooses a suitable template to render the content (surface realization)—e.g., *winds between* [min] *and* [max] *mph*; templates are automatically extracted from training data.

We tested our approach in three domains: ROBOCUP, for sportscasting (Chen and Mooney, 2008); SUMTIME, for technical weather forecast generation (Reiter et al., 2005); and WEATHERGOV, for common weather forecast generation (Liang et al., 2009). We performed both automatic (BLEU) and human evaluation. On WEATHERGOV, we

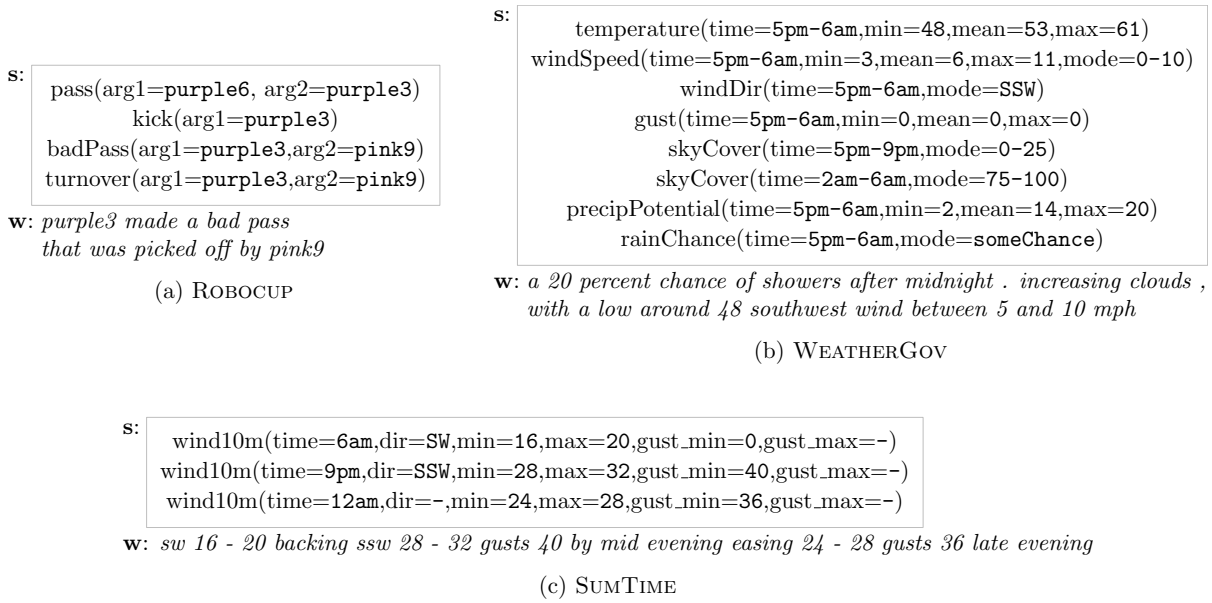

Figure 1: Example scenarios (a scenario is a world state $\mathbf{s}$ paired with a text $\mathbf{w}$) for each of the three domains. Each row in the world state denotes a record. Our generation task is to map a world state $\mathbf{s}$ (input) to a text $\mathbf{w}$ (output). Note that this mapping involves both content selection and surface realization.

achieved a BLEU score of 51.5 on the combined task of content selection and generation, which is more than a two-fold improvement over a model similar to that of Liang et al. (2009). On ROBOCUP and SUMTIME, we achieved results comparable to the state-of-the-art. most importantly, we obtained these results with a general-purpose approach that we believe is simpler than current state-of-the-art systems.

2 Setup and Domains

Our goal is to generate a text given a world state. The *world state*, denoted $\mathbf{s}$, is represented by a set of database records. Define $\mathcal{T}$ to be a set of record types, where each record type $t \in \mathcal{T}$ is associated with a set of fields $\text{FIELDS}(t)$. Each *record* $r \in \mathbf{s}$ has a record type $r.t \in \mathcal{T}$ and a *field value* $r.v[f]$ for each field $f \in \text{FIELDS}(t)$. The *text*, denoted $\mathbf{w}$, is represented by a sequence of tokenized words. We use the term *scenario* to denote a world state $\mathbf{s}$ paired with a text $\mathbf{w}$.

In this paper, we conducted experiments on three domains, which are detailed in the following subsections. Example scenarios for each domain are detailed in Figure 1.

2.1 ROBOCUP: Sportscasting

A world state in the ROBOCUP domain is a set of event records (*meaning representations* in the terminology of Chen and Mooney (2008)) generated by a robot soccer simulator. For example, the record pass(arg1=`pink1`,arg2=`pink5`) denotes a passing event; records of this type (pass) have two fields: arg1 (the agent) and arg2 (the recipient). As the game progresses, human commentators talk about some of the events in the game, e.g., *purple3 made a bad pass that was picked off by pink9*.

We used the dataset created by Chen and Mooney (2008), which contains 1919 scenarios from the 2001–2004 Robocup finals. Each scenario consists of a single sentence representing a fragment of a commentary on the game, paired with a set of candidate records, which were recorded within five seconds of the commentary. The records in the ROBOCUP dataset data were aligned by Chen and Mooney (2008). Each scenario contains on average $|\mathbf{s}| = 2.4$ records and 5.7 words. See Figure 1(a) for an example of a scenario. Content selection in this domain is choosing the single record to talk about, and surface realization is talking about it.

2.2 SUMTIME: Technical Weather Forecasts

Reiter et al. (2005) developed a generation system and created the SUMTIME-METEO corpus, which consists of marine wind weather forecasts used by offshore oil rigs, generated by the output of weather simulators. More specifically, these forecasts describe various aspects of the wind at different times during the forecast period.

We used the version of the SUMTIME-METEO corpus created by Belz (2008). The dataset consists of 469 scenarios, each containing on average $|\mathbf{s}| = 2.6$ records and 16.2 words. See Figure 1(c) for an example of a scenario. This task requires no content selection, only surface realization: The records are given in some fixed order and the task is to generate from each of these records in turn; of course, due to contextual dependencies, these records cannot be generated independently.

2.3 WEATHERGOV: Common Weather Forecasts

In the WEATHERGOV domain, the world state contains detailed information about a local weather forecast (e.g., temperature, rain chance, etc.). The text is a short forecast report based on this information.

We used the dataset created by Liang et al. (2009). The world state is summarized by records which aggregate measurements over selected time intervals. The dataset consists of 29,528 scenarios, each containing on average $|\mathbf{s}| = 36$ records and 28.7 words. See Figure 1(b) for an example of a scenario.

While SUMTIME and WEATHERGOV are both weather domains, there are significant differences between the two. SUMTIME forecasts are intended to be read by trained meteorologists, and thus the text is quite abbreviated. On the other hand, WEATHERGOV texts are intended to be read by the general public and thus is more English-like. Furthermore, SUMTIME does not require content selection, whereas content selection is a major focus of WEATHERGOV. Indeed, on average, only 5 of 36 records are actually mentioned in a WEATHERGOV scenario. Also, WEATHERGOV is more complex: The text is more varied, there are multiple record types, and there are about ten times as many records in each world state.

Generation Process

for $i = 1, 2, \ldots$:
  choose a record $r_i \in \mathbf{s}$
  if $r_i =$ STOP: return
  choose a field set $F_i \subset \text{FIELDS}(r_i.t)$
  choose a template $T_i \in \text{TEMPLATES}(r_i.t, F_i)$

Figure 2: Pseudocode for the generation process. The generated text $\mathbf{w}$ is a deterministic function of the decisions.

3 The Generation Process

To model the process of generating a text $\mathbf{w}$ from a world state $\mathbf{s}$, we decompose the generation process into a sequence of local decisions. There are two aspects of this decomposition that we need to specify: (i) how the decisions are structured; and (ii) what pieces of information govern the decisions.

The decisions are structured hierarchically into three types of decisions: (i) *record decisions*, which determine which records in the world state to talk about (macro content selection); (ii) *field set decisions*, which determine which fields of those records to mention (micro content selection); and (iii) *template decisions*, which determine the actual words to use to describe the chosen fields (surface realization). Figure 2 shows the pseudocode for the generation process, while Figure 3 depicts an example of the generation process on a WEATHERGOV scenario.

Each of these decisions is governed by a set of feature templates (see Figure 4), which are represented as functions of the current decision and past decisions. The feature weights are learned from training data (see Section 4.3).

We chose a set of generic domain-independent feature templates, described in the sections below. These features can, in general, depend on the current decision and *all* previous decisions. For example, referring to Figure 4, **R2** features on the record choice depend on all the previous record decisions, and **R5** features depend on the most recent template decision. This is in contrast with most systems for content selection (Barzilay and Lee, 2004) and surface realization (Belz, 2008), where decisions must decompose locally according to either a graph or tree. The ability to use global features in this manner is

Figure 3: The generation process on an example WEATHERGOV scenario. The figure is divided into two parts: The upper part of the figure shows the generation of text from the world state via a sequence of seven decisions (in boxes). Three of these decisions are highlighted and the features that govern these decisions are shown in the lower part of the figure. Note that different decisions in the generation process would result in different features being active (nonzero).

Feature Templates

Record	**R1**†	list of last k record types	⟦$r_i.t = *$ and ($r_{i-1}.t, \ldots, r_{i-k}.t$) = $*$⟧ for $k \in \{1, 2\}$
	R2	set of previous record types	⟦$r_i.t = *$ and {$r_j.t : j < i$} = $*$⟧
	R3	record type already generated	⟦$r_j.t = r_i.t$ for some $j < i$⟧
	R4	field values	⟦$r_i.t = *$ and $r_i.v[f] = *$⟧ for $f \in$ FIELDS($r_i.t$)
	R5†	stop under language model (LM)	⟦$r_i.t$ = STOP⟧ × log p_{LM}(STOP | previous two words generated)
Field set	**F1**†	field set	⟦$F_i = *$⟧
	F2	field values	⟦$F_i = *$ and $r_i.v[f] = *$⟧ for $f \in F_i$
Template	**W1**†	base/coarse generation template	⟦$h(T_i) = *$⟧ for $h \in$ {BASE, COARSE}
	W2	field values	⟦$h(T_i) = *$ and $r_i.v[f] = *$⟧ for $f \in F_i, h \in$ {BASE, COARSE}
	W3†	first word of template under LM	log p_{LM}(first word in T_i | previous two words)

Figure 4: Feature templates that govern the record, field set, and template decisions. Each line specifies the name, informal description, and formal description of a set of features, obtained by ranging $*$ over possible values (for example, for ⟦$r_i.t = *$⟧, $*$ ranges over all record types $\mathcal{T}$). Notation: ⟦e⟧ returns 1 if the expression e is true and 0 if it is false. These feature templates are domain-independent; that is, they are used to create features automatically across domains. Feature templates marked with † are included in our baseline system (Section 5.2).

one of the principal advantages of our approach.

3.1 Record Decisions

Record decisions are responsible for macro content selection. Each record decision chooses a record r_i from the world state $\mathbf{s}$ according to features of the following types:

R1 captures the discourse coherence aspect of content selection; for example, we learn that windSpeed tends to follow windDir (but not al-

ways). **R2** captures an unordered notion of coherence—simply which sets of record types are preferable; for example, we learn that rainChance is not generated if sleetChance already was mentioned. **R3** is a coarser version of **R2**, capturing how likely it is to propose a record of a type that has already been generated. **R4** captures the important aspect of content selection that the records chosen depend on their field values;[1] for example, we learn that snowChance is not chosen unless there is snow. **R5** allows the language model to indicate whether a STOP record is appropriate; this helps prevent sentences from ending abruptly.

3.2 Field Set Decisions

Field set decisions are responsible for micro content selection, i.e., which fields of a record are mentioned. Each field set decision chooses a subset of fields F_i from the set of fields $\textsc{Fields}(r_i.t)$ of the record r_i that was just generated. These decisions are made based on two types of features:

F1 captures which sets of fields are talked about together; for example, we learn that $\{\text{mean}\}$ and $\{\text{min}, \text{max}\}$ are preferred field sets for the windSpeed record. By defining features on the entire field set, we can capture any correlation structure over the fields; in contrast, Liang et al. (2009) generates a sequence of fields in which a field can only depend on the previous one.

F2 allows the field set to be chosen based on the values of the fields, analogously to **R4**.

3.3 Template Decisions

Template decisions perform surface realization. A template is a sequence of elements, where each element is either a word (e.g., *around*) or a field (e.g., [min]). Given the record r_i and field set F_i that we are generating from, the goal is to choose a template T_i (Section 4.3.2 describes how we define the set of possible templates). The features that govern the choice of T_i are as follows:

W1 captures a priori preferences for generation templates given field sets. There are two ways to control this preference, BASE and COARSE. $\textsc{Base}(T_i)$ denotes the template T_i itself, thus allowing us to remember exactly which templates were useful. To guard against overfitting, we also use $\textsc{Coarse}(T_i)$, which maps T_i to a coarsened version of T_i, in which more words are replaced with their associated fields (see Figure 5 for an example).

W2 captures a dependence on the values of fields in the field set, and is analogous to **R4** and **F2**. Finally, **W3** contributes a language model probability, to ensure smooth transitions between templates.

After T_i has been chosen, each field in the template is replaced with a word given the corresponding field value in the world state. In particular, a word is chosen from the parameters learned in the model of Liang et al. (2009). In the example in Figure 3, the [min] field in T_2 has value 44, which is rendered to the word *45* (rounding and other noisy deviations are common in the WEATHERGOV domain).

4 Learning a Probabilistic Model

Having described all the features, we now present a conditional probabilistic model over texts $\mathbf{w}$ given world states $\mathbf{s}$ (Section 4.1). Section 4.2 describes how to use the model for generation, and Section 4.3 describes how to learn the model.

4.1 Model

Recall from Section 3 that the generation process generates $r_1, F_1, T_1, r_2, F_2, T_2, \ldots, \text{STOP}$. To unify notation, denote this sequence of decisions as $\mathbf{d} = (d_1, \ldots, d_{|\mathbf{d}|})$.

Our probability model is defined as follows:

$$p(\mathbf{d} \mid \mathbf{s}; \theta) = \prod_{j=1}^{|\mathbf{d}|} p(d_j \mid \mathbf{d}_{<j}; \theta), \qquad (1)$$

where $\mathbf{d}_{<j} = (d_1, \ldots, d_{j-1})$ is the history of decisions and θ are the model parameters (feature weights). Note that the text $\mathbf{w}$ (the output) is a deterministic function of the decisions $\mathbf{d}$. We use the features described in Section 3 to define a log-linear model for each decision:

$$p(d_j \mid \mathbf{d}_{<j}, \mathbf{s}; \theta) = \frac{\exp\{\phi_j(d_j, \mathbf{d}_{<j}, \mathbf{s})^\top \theta\}}{\sum_{d'_j \in \mathcal{D}_j} \exp\{\phi_j(d'_j, \mathbf{d}_{<j}, \mathbf{s})^\top \theta\}}, \quad (2)$$

where θ are all the parameters (feature weights), ϕ_j is the feature vector for the j-th decision, and $\mathcal{D}_j$ is

[1] We map a numeric field value onto one of five categories (`very-low`, `low`, `medium`, `high`, or `very-high`) based on its value with respect to the mean and standard deviation of values of that field in the training data.

the domain of the j-th decision (either records, field sets, or templates).

This chaining of log-linear models was used in Ratnaparkhi (1998) for tagging and parsing, and in Ratnaparkhi (2002) for surface realization. The ability to condition on arbitrary histories is a defining property of these models.

4.2 Using the Model for Generation

Suppose we have learned a model with parameters θ (how to obtain θ is discussed in Section 4.3). Given a world state $\mathbf{s}$, we would like to use our model to generate an output text $\mathbf{w}$ via a decision sequence $\mathbf{d}$.

In our experiments, we choose $\mathbf{d}$ by sequentially choosing the best decision in a greedy fashion (until the STOP record is generated):

$$d_j = \operatorname*{argmax}_{d'_j} p(d'_j \mid \mathbf{d}_{<j}, \mathbf{s}; \theta). \quad (3)$$

Alternatively, instead of choosing the best decision at each point, we can sample from the distribution: $d_j \sim p(d_j \mid \mathbf{d}_{<j}, \mathbf{s}; \theta)$, which provides more diverse generated texts at the expense of a slight degradation in quality.

Both greedy search and sampling are very efficient. Another option is to try to find the Viterbi decision sequence, i.e., the one with the maximum joint probability: $\mathbf{d} = \operatorname{argmax}_{\mathbf{d}'} p(\mathbf{d}' \mid \mathbf{s}; \theta)$. However, this computation is intractable due to features depending arbitrarily on past decisions, making dynamic programming infeasible. We tried using beam search to approximate this optimization, but we actually found that beam search performed worse than greedy. Belz (2008) also found that greedy was more effective than Viterbi for their model.

4.3 Learning

Now we turn our attention to learning the parameters θ of our model. We are given a set of N scenarios $\{(\mathbf{s}^{(i)}, \mathbf{w}^{(i)})\}_{i=1}^{N}$ as training data. Note that our model is defined over the decision sequence $\mathbf{d}$ which contains information not present in $\mathbf{w}$. In Sections 4.3.1 and 4.3.2, we show how we fill in this missing information to obtain $\mathbf{d}^{(i)}$ for each training scenario i.

Assuming this missing information is filled, we end up with a standard supervised learning problem, which can be solved by maximize the (conditional) likelihood of the training data:

$$\max_{\theta \in \mathbb{R}^d} \left(\sum_{i=1}^{N} \sum_{j=1}^{|\mathbf{d}^{(i)}|} \log p(d_j^{(i)} \mid \mathbf{d}_{<j}^{(i)}; \theta) \right) - \lambda ||\theta||^2, \quad (4)$$

where $\lambda > 0$ is a regularization parameter. The objective function in (4) is optimized using the standard L-BFGS algorithm (Liu and Nocedal, 1989).

4.3.1 Latent Alignments

As mentioned previously, our training data includes only the world state $\mathbf{s}$ and generated text $\mathbf{w}$, not the full sequence of decisions $\mathbf{d}$ needed for training. Intuitively, we know *what* was generated but not *why* it was generated.

We use the model of Liang et al. (2009) to impute the decisions $\mathbf{d}$. They introduce a generative model $p(\mathbf{a}, \mathbf{w}|\mathbf{s})$, where the latent *alignment* $\mathbf{a}$ specifies (1) the sequence of records that were chosen, (2) the sequence of fields that were chosen, and (3) which words in the text were spanned by the chosen records and fields. The model is learned in an unsupervised manner using EM to produce $\mathbf{a}$ observing only $\mathbf{w}$ and $\mathbf{s}$.

An example of an alignment is given in the left part of Figure 5. This information specifies the record decisions and a set of fields for each record. Because the induced alignments can be noisy, we need to process them to obtain cleaner template decisions. This is the subject of the next section.

4.3.2 Template Extraction

Given an aligned training scenario (Figure 5), we would like to extract two types of templates.

For each record, an aligned training scenario specifies a sequence of fields and the text that is spanned by each field. We create a template by abstracting fields—that is, replacing the words spanned by a field by the field itself. We call the resulting template COARSE. The problem with using this template directly is that fields can be noisy due to errors from the unsupervised model.

Therefore, we also create a BASE template which only abstracts a subset of the fields. In particular, we define a *trigger pattern* which specifies a simple condition under which a field should be abstracted. For WEATHERGOV, we only abstract fields that

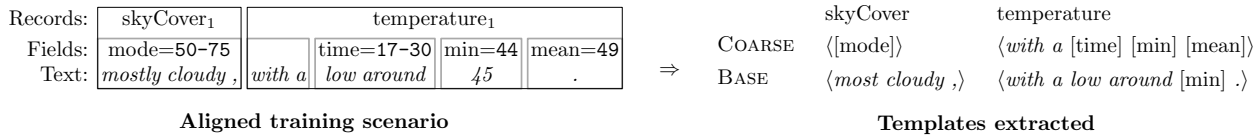

Figure 5: An example of template extraction from an imperfectly aligned training scenario. Note that these alignments are noisy (e.g., [mean] aligns to a period). Therefore, for each record (skyCover and temperature in this case), we extract two templates: (1) a COARSE template, which takes the text spanned by the record and abstracts away all fields in the scenario ([mode], [time], [min], and [mean] in the example); and (2) a BASE template, which only abstracts away fields whose spanned text matches a simple pattern (e.g., numbers in WEATHERGOV, corresponding to [min] in the example).

span numbers; for SUMTIME, fields that span numbers and wind directions; and for ROBOCUP, fields that span words starting with *purple* or *pink*.

For each record r_i, we define T_i so that BASE(T_i) and COARSE(T_i) are the corresponding two extracted templates. We restrict F_i to the set of abstracted fields in the COARSE template

5 Experiments

We now present an empirical evaluation of our system on our three domains—ROBOCUP, SUMTIME, and WEATHERGOV.

5.1 Evaluation Metrics

Automatic Evaluation To evaluate surface realization (or, combined content selection and surface realization), we measured the BLEU score (Papineni et al., 2002) (the precision of 4-grams with a brevity penalty) of the system-generated output with respect to the human-generated output.

To evaluate macro content selection, we measured the F_1 score (the harmonic mean of precision and recall) of the set of records chosen with respect to the human-annotated set of records.

Human Evaluation We conducted a human evaluation using Amazon Mechanical Turk. For each domain, we chose 100 scenarios randomly from the test set. We ran each system under consideration on each of these scenarios, and presented each resulting output to 10 evaluators.[2] Evaluators were given instructions to rank an output on the basis of English fluency and semantic correctness on the following scale:

Score	English Fluency	Semantic Correctness
5	Flawless	Perfect
4	Good	Near Perfect
3	Non-native	Minor Errors
2	Disfluent	Major Errors
1	Gibberish	Completely Wrong

Evaluators were also given additional domain-specific information: (1) the background of the domain (e.g., that SUMTIME reports are technical weather reports); (2) general properties of the desired output (e.g., that SUMTIME texts should mention every record whereas WEATHERGOV texts need not); and (3) peculiarities of the text (e.g., the suffix *ly* in SUMTIME should exist as a separate token from its stem, or that *pink goalie* and *pink1* have the same meaning in ROBOCUP).

5.2 Systems

We evaluated the following systems on our three domains:

- HUMAN is the human-generated output.
- OURSYSTEM uses all the features in Figure 4 and is trained according to Section 4.3.
- BASELINE is OURSYSTEM using a subset of the features (those marked with † in Figure 4). In contrast to OURSYSTEM, the included features only depend on a local context of decisions in a manner similar to the generative model of Liang et al. (2009) and the *p*CRU-greedy system of Belz (2008). BASELINE also excludes features that depend on values of the world state.
- The existing state-of-the-art domain-specific system for each domain.

5.3 ROBOCUP Results

Following the evaluation methodology of Chen and Mooney (2008), we trained our system on three

[2]To minimize bias, we evaluated all the systems at once, randomly shuffling the outputs of the systems. The evaluators were not necessarily the same 10 evaluators.

System	F_1	BLEU*	English Fluency	Semantic Correctness
BASELINE	78.7	24.8	4.28 ± 0.78	4.15 ± 1.14
OURSYSTEM	79.9	28.8	4.34 ± 0.69	4.17 ± 1.21
WASPER-GEN	72.0	28.7	4.43 ± 0.76	4.27 ± 1.15
HUMAN	—	—	4.43 ± 0.69	4.30 ± 1.07

Table 1: ROBOCUP results. WASPER-GEN is described in Chen and Mooney (2008). The BLEU is reported on systems that use fixed human-annotated records (in other words, we evaluate surface realization given perfect content selection).

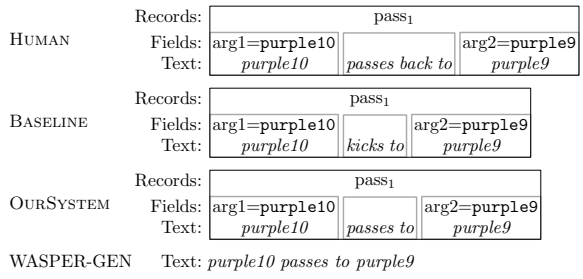

Figure 6: Outputs of systems on an example ROBOCUP scenario. There are some minor differences between the outputs. Recall that OURSYSTEM differs from BASELINE mostly in the addition of feature **W2**, which captures dependencies between field values (e.g., `purple10`) and the template chosen (e.g., [arg1] *passes to* [arg2]). This allows us to capture value-dependent preferences for different realizations (e.g., *passes to* over *kicks to*). Also, HUMAN uses *passes back to*, but this word choice requires knowledge of passing records in previous scenarios, which none of the systems have access to. It would natural, however, to add features that would capture these longer-range dependencies in our framework.

Robocup games and tested on the fourth, averaging over the four train/test splits. We report the average test accuracy weighted by the number of scenarios in a game. First, we evaluated macro content selection. Table 1 shows that OURSYSTEM significantly outperforms BASELINE and WASPER-GEN on F_1.

To compare with Chen and Mooney (2008) on surface realization, we fixed each system's record decisions to the ones given by the annotated data and enforced that all the fields of that record are chosen. Table 1 shows that OURSYSTEM significantly outperforms BASELINE and is comparable to WASPER-GEN on BLEU. On human evaluation, OURSYSTEM outperforms BASELINE, but WASPER-GEN outperforms OURSYSTEM. See Figure 6 for example outputs from the various systems.

	BLEU	English Fluency	Semantic Correctness
BASELINE	32.9	4.23 ± 0.71	4.26 ± 0.85
OURSYSTEM	55.1	4.25 ± 0.69	4.27 ± 0.82
OURSYSTEM-CUSTOM	62.3	4.12 ± 0.78	4.33 ± 0.91
*p*CRU-greedy	63.6	4.18 ± 0.71	4.49 ± 0.73
SUMTIME-Hybrid	52.7	—	—
HUMAN	—	4.09 ± 0.83	4.37 ± 0.87

Table 2: SUMTIME results. The SUMTIME-Hybrid system is described in (Reiter et al., 2005); *p*CRU-greedy, in (Belz, 2008).

5.4 SUMTIME Results

The SUMTIME task only requires micro content selection and surface realization because the sequence of records to be generated is fixed; only these aspects are evaluated. Following the methodology of Belz (2008), we used five-fold cross validation.

We found that using the unsupervised model of Liang et al. (2009) to automatically produce aligned training scenarios (Section 4.3.1) was less effective than it was in the other two domains due to two factors: (i) there are fewer training examples in SUMTIME and unsupervised learning typically works better with a large amount of data; and (ii) the alignment model does not exploit the temporal structure in the SUMTIME world state. Therefore, we used a small set of simple regular expressions to produce aligned training scenarios.

Table 2 shows that OURSYSTEM significantly outperforms BASELINE as well as SUMTIME-Hybrid, a hand-crafted system, on BLEU. Note that OURSYSTEM is domain-independent and has not been specifically tuned to SUMTIME. However, OURSYSTEM is outperformed by the state-of-the-art statistical system *p*CRU-greedy.

Custom Features One of the advantages of our feature-based approach is that it is straightforward to incorporate domain-specific features to capture specific properties of a domain. To this end, we define the following set of feature templates in place of our generic feature templates from Figure 4:

- **F1′**: Value of time
- **F2′**: Existence of gusts/wind direction/wind speeds
- **W1′**: Change in wind direction (clockwise, counterclockwise, or none)

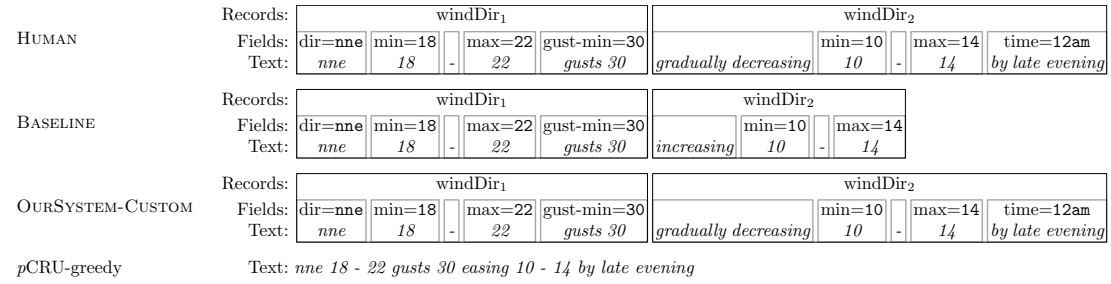

Figure 7: Outputs of systems on an example SUMTIME scenario. Two notable differences between OURSYSTEM-CUSTOM and BASELINE arise due to OURSYSTEM-CUSTOM's value-dependent features. For example, OURSYSTEM-CUSTOM can choose whether to include the time field (windDir_2) or not (windDir_1), depending on the value of the time (**F1′**), thereby improving content selection. OURSYSTEM-CUSTOM also improves surface realization, choosing *gradually decreasing* over BASELINE's *increasing*. Interestingly, this improvement comes from the joint effort of two features: **W2′** prefers *decreasing* over *increasing* in this case, and **W5′** adds the modifier *gradually*. An important strength of log-linear models is the ability to combine soft preferences from many features.

- **W2′**: Change in wind speed
- **W3′**: Change in wind direction and speed
- **W4′**: Existence of gust min and/or max
- **W5′**: Time elapsed since last record
- **W6′**: Whether wind is a cardinal direction (N, E, S, W)

The resulting system, which we call OURSYSTEM-CUSTOM, obtains a BLEU score which is comparable to *p*CRU-greedy.

An important aspect of our system that it is flexible and quick to deploy. According to Belz (2008), SUMTIME-Hybrid took twelve person-months to build, while *p*CRU-greedy took one month. Having developed OURSYSTEM in a domain-independent way, we only needed to do simple reformatting upon receiving the SUMTIME data. Furthermore, it took only a few days to develop the custom features above to create OURSYSTEM-CUSTOM, which has BLEU performance comparable to the state-of-the-art *p*CRU-greedy system.

We also conducted human evaluations on the four systems shown in Table 2. Note that this evaluation is rather difficult for Mechanical Turkers since SUMTIME texts are rather technical compared to those in other domains. Interestingly, all systems outperform HUMAN on English fluency; this result corroborates the findings of Belz (2008). On semantic correctness, all systems perform comparably to HUMAN, except *p*CRU-greedy, which performs slightly better. See Figure 7 for a comparison of the outputs generated by the various systems.

	F_1	BLEU*	English Fluency	Semantic Correctness
BASELINE	22.1	22.2	4.07 ± 0.59	3.41 ± 1.16
OURSYSTEM	65.4	51.5	4.12 ± 0.74	4.22 ± 0.89
HUMAN	—	—	4.14 ± 0.71	3.85 ± 0.99

Table 3: WEATHERGOV results. The BLEU score is on joint content selection and surface realization and is modified to not penalize numeric deviations of at most 5.

5.5 WEATHERGOV Results

We evaluate the WEATHERGOV corpus on the joint task of content selection and surface realization. We split our corpus into 25,000 scenarios for training, 1,000 for development, and 3,528 for testing. In WEATHERGOV, numeric field values are often rounded or noisily perturbed, so it is difficult to generate precisely matching numbers. Therefore, we used a modified BLEU score where numbers differing by at most five are treated as equal. Furthermore, WEATHERGOV is evaluated on the joint content selection and surface realization task, unlike ROBOCUP, where content selection and surface realization were treated separately, and SUMTIME, where content selection was not applicable.

Table 3 shows the results. We see that OURSYSTEM substantially outperforms BASELINE, especially on BLEU score and semantic correctness. This difference shows that taking non-local context into account is very important in this domain. This result is not surprising, since WEATHERGOV is the most complicated of the three domains, and this complexity is exactly where non-locality is neces-

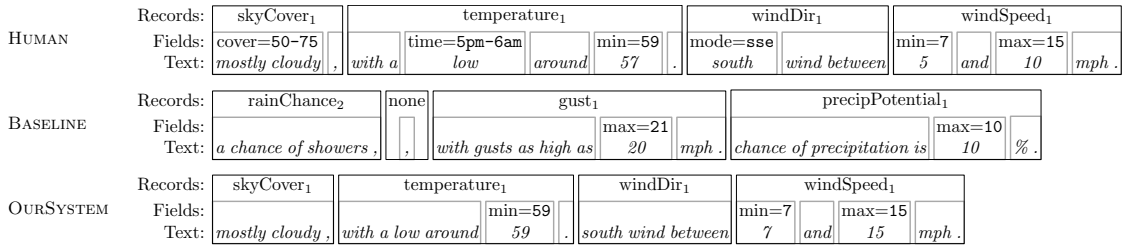

Figure 8: Outputs of systems on an example WEATHERGOV scenario. Most of the gains of OURSYSTEM over BASELINE come from improved content selection. For example, BASELINE chooses rainChance because it happens to be the most common first record type in the training data. However, since OURSYSTEM has features that depend on the value of rainChance (`noChance` in this case), it has learned to disprefer talking about rain when there is no rain. Also, OURSYSTEM has additional features on the entire history of chosen records, which enables it to choose a better sequence of records.

sary. Interestingly, OURSYSTEM even outperforms HUMAN on semantic correctness, perhaps due to generating more straightforward renderings of the world state. Figure 8 describes example outputs for each system.

6 Related Work

There has been a fair amount of work both on content selection and surface realization. In content selection, Barzilay and Lee (2004) use an approach based on local classification with edge-wise scores between local decisions. Our model, on the other hand, can capture higher-order constraints to enforce global coherence.

Liang et al. (2009) introduces a generative model of the text given the world state, and in some ways is similar in spirit to our model. Although that model is capable of generation in principle, it was designed for unsupervised induction of hidden alignments (which is exactly what we use it for). Even if combined with a language model, generated text was much worse than our baseline.

The prominent approach for surface realization is rendering the text from a grammar. Wong and Mooney (2007) and Chen and Mooney (2008) use synchronous grammars that map a logical form, represented as a tree, into a parse of the text. Soricut and Marcu (2006) uses tree structures called WIDL-expressions (the acronym corresponds to four operations akin to the rewrite rules of a grammar) to represent the realization process, and, like our approach, operates in a log-linear framework. Belz (2008) and Belz and Kow (2009) also perform surface realization from a PCFG-like grammar. Lu et al. (2009) uses a conditional random field model over trees. Other authors have performed surface realization using various grammar formalisms, for instance CCG (White et al., 2007), HPSG (Nakanishi et al., 2005), and LFG (Cahill and van Genabith, 2006).

In each of the above cases, the decomposable structure of the tree/grammar enables tractability. However, we saw that it was important to include features that captured long-range dependencies. Our model is also similar in spirit to Ratnaparkhi (2002) in the use of non-local features, but we operate at three levels of hierarchy to include both content selection and surface realization.

One issue that arises with long-range dependencies is the lack of efficient algorithms for finding the optimal text. Koller and Striegnitz (2002) perform surface realization of a flat semantics, which is NP-hard, so they recast the problem as non-projective dependency parsing. Ratnaparkhi (2002) uses beam search to find an approximate solution. We found that a greedy approach obtained better results than beam search; Belz (2008) found greedy approaches to be effective as well.

7 Conclusion

We have developed a simple yet powerful generation system that combines both content selection and surface realization in a domain independent way. Despite our approach being domain-independent, we were able to obtain performance comparable to the state-of-the-art across three domains. Additionally, the feature-based design of our approach makes it easy to incorporate domain-specific knowledge to increase performance even further.

References

R. Barzilay and L. Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *Human Language Technology and North American Association for Computational Linguistics (HLT/NAACL)*.

A. Belz and E. Kow. 2009. System building cost vs. output quality in data-to-text generation. In *European Workshop on Natural Language Generation*, pages 16–24.

A. Belz. 2008. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Natural Language Engineering*, 14(4):1–26.

Aoife Cahill and Josef van Genabith. 2006. Robust pcfg-based generation using automatically acquired LFG approximations. In *Association for Computational Linguistics (ACL)*, pages 1033–1040, Morristown, NJ, USA. Association for Computational Linguistics.

D. L. Chen and R. J. Mooney. 2008. Learning to sportscast: A test of grounded language acquisition. In *International Conference on Machine Learning (ICML)*, pages 128–135.

R. Dale, S. Geldof, and J. Prost. 2003. Coral: using natural language generation for navigational assistance. In *Australasian computer science conference*, pages 35–44.

M. E. Foster and M. White. 2004. Techniques for text planning with XSLT. In *Workshop on NLP and XML: RDF/RDFS and OWL in Language Technology*, pages 1–8.

N. Green. 2006. Generation of biomedical arguments for lay readers. In *International Natural Language Generation Conference*, pages 114–121.

A. Koller and K. Striegnitz. 2002. Generation as dependency parsing. In *Association for Computational Linguistics (ACL)*, pages 17–24.

P. Liang, M. I. Jordan, and D. Klein. 2009. Learning semantic correspondences with less supervision. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

D. C. Liu and J. Nocedal. 1989. On the limited memory method for large scale optimization. *Mathematical Programming B*, 45(3):503–528.

W. Lu, H. T. Ng, and W. S. Lee. 2009. Natural language generation with tree conditional random fields. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 400–409.

Hiroko Nakanishi, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic models for disambiguation of an HPSG-based chart generator. In *Parsing '05: Proceedings of the Ninth International Workshop on Parsing Technology*, pages 93–102, Morristown, NJ, USA. Association for Computational Linguistics.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Association for Computational Linguistics (ACL)*.

A. Ratnaparkhi. 1998. *Maximum entropy models for natural language ambiguity resolution*. Ph.D. thesis, University of Pennsylvania.

A. Ratnaparkhi. 2002. Trainable approaches to surface natural language generation and their application to conversational dialog systems. *Computer, Speech & Language*, 16:435–455.

E. Reiter, S. Sripada, J. Hunter, J. Yu, and I. Davy. 2005. Choosing words in computer-generated weather forecasts. *Artificial Intelligence*, 167:137–169.

R. Soricut and D. Marcu. 2006. Stochastic language generation using WIDL-expressions and its application in machine translation and summarization. In *Association for Computational Linguistics (ACL)*, pages 1105–1112.

R. Turner, Y. Sripada, and E. Reiter. 2009. Generating approximate geographic descriptions. In *European Workshop on Natural Language Generation*, pages 42–49.

Michael White, Rajakrishnan Rajkumar, and Scott Martin. 2007. Towards broad coverage surface realization with CCG. In *In Proceedings of the Workshop on Using Corpora for NLG: Language Generation and Machine Translation (UCNLG+MT)*.

Y. W. Wong and R. J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Association for Computational Linguistics (ACL)*, pages 960–967.