Natural Language Generation with Vocabulary Constraints

Ben Swanson
Brown University
Providence, RI
chonger@cs.brown.edu

Elif Yamangil
Google Inc.
Mountain View, CA
leafer@google.com

Eugene Charniak
Brown University
Providence, RI
ec@cs.brown.edu

Abstract

We investigate data driven natural language generation under the constraints that all words must come from a fixed vocabulary and a specified word must appear in the generated sentence, motivated by the possibility for automatic generation of language education exercises. We present fast and accurate approximations to the ideal rejection samplers for these constraints and compare various sentence level generative language models. Our best systems produce output that is with high frequency both novel and error free, which we validate with human and automatic evaluations.

1 Introduction

Freeform data driven Natural Language Generation (NLG) is a topic explored by academics and artists alike, but motivating its empirical study is a difficult task. While many language models used in statistical NLP are generative and can easily produce sample sentences by running their "generative mode", if all that is required is a plausible sentence one might as well pick a sentence at random from any existing corpus.

NLG becomes useful when constraints exist such that only certain sentences are valid. The majority of NLG applies a semantic constraint of "what to say", producing sentences with communicative goals. Other work such as ours investigates constraints in structure; producing sentences of a certain form without concern for their specific meaning.

We study two constraints concerning the words that are allowed in a sentence. The first sets a fixed vocabulary such that only sentences where all words are in-vocab are allowed. The second demands not only that all words are in-vocab, but also requires the inclusion of a specific word somewhere in the sentence.

These constraints are natural in the construction of language education exercises, where students have small known vocabularies and exercises that reinforce the knowledge of arbitrary words are required. To provide an example, consider a Chinese teacher composing a quiz that asks students to translate sentences from English to Chinese. The teacher cannot ask students to translate words that have not been taught in class, and would like ensure that each vocabulary word from the current book chapter is included in at least one sentence. Using a system such as ours, she could easily generate a number of usable sentences that contain a given vocab word and select her favorite, repeating this process for each vocab word until the quiz is complete.

The construction of such a system presents two primary technical challenges. First, while highly parameterized models trained on large corpora are a good fit for data driven NLG, sparsity is still an issue when constraints are introduced. Traditional smoothing techniques used for prediction based tasks are inappropriate, however, as they liberally assign probability to implausible text. We investigate smoothing techniques better suited for NLG that smooth more precisely, sharing probability only between words that have strong semantic connections.

The second challenge arises from the fact that both vocabulary and word inclusion constraints are easily handled with a rejection sampler that repeatedly generates sentences until one that obeys the constraints is produced. Unfortunately, for

models with a sufficiently wide range of outputs the computation wasted by rejection quickly becomes prohibitive, especially when the word inclusion constraint is applied. We define models that sample directly from the possible outputs for each constraint without rejection or backtracking, and closely approximate the distribution of the true rejection samplers.

We contrast several generative systems through both human and automatic evaluation. Our best system effectively captures the compositional nature of our training data, producing error-free text with nearly 80 percent accuracy without wasting computation on backtracking or rejection. When the word inclusion constraint is introduced, we show clear empirical advantages over the simple solution of searching a large corpus for an appropriate sentence.

2 Related Work

The majority of NLG focuses on the satisfaction of a communicative goal, with examples such as Belz (2008) which produces weather reports from structured data or Mitchell et al. (2013) which generates descriptions of objects from images. Our work is more similar to NLG work that concentrates on structural constraints such as generative poetry (Greene et al., 2010) (Colton et al., 2012) (Jiang and Zhou, 2008) or song lyrics (Wu et al., 2013) (Ramakrishnan A et al., 2009), where specified meter or rhyme schemes are enforced. In these papers soft semantic goals are sometimes also introduced that seek responses to previous lines of poetry or lyric.

Computational creativity is another subfield of NLG that often does not fix an a priori meaning in its output. Examples such as Özbal et al. (2013) and Valitutti et al. (2013) use template filling techniques guided by quantified notions of humor or how catchy a phrase is.

Our motivation for generation of material for language education exists in work such as Sumita et al. (2005) and Mostow and Jang (2012), which deal with automatic generation of classic fill in the blank questions. Our work is naturally complementary to these efforts, as their methods require a corpus of in-vocab text to serve as seed sentences.

3 Freeform Generation

For clarity in our discussion, we phrase the sentence generation process in the following general terms based around two classes of atomic units : *contexts* and *outcomes*. In order to specify a generation system, we must define

1. the set $\mathcal{C}$ of contexts c
2. the set $\mathcal{O}$ of outcomes o
3. the "Imply" function $I(c, o) \rightarrow List[c \in \mathcal{C}]$
4. $\mathcal{M}$: derivation tree $\rightleftarrows$ sentence

where $I(c, o)$ defines the further contexts implied by the choice of outcome o for the context c. Beginning with a unique root context, a derivation tree is created by repeatedly choosing an outcome o for a leaf context c and expanding c to the new leaf contexts specified by $I(c, o)$. $\mathcal{M}$ converts between derivation tree and sentence text form.

This is simply a convenient rephrasing of the Context Free Grammar formalism, and as such the systems we describe all have some equivalent CFG interpretation. Indeed, to describe a traditional CFG, let $\mathcal{C}$ be the set of symbols, $\mathcal{O}$ be the rules of the CFG, and $I(c, o)$ return a list of the symbols on the right hand side of the rule o. To define an n-gram model, a context is a list of words, an outcome a single word, and $I(c, o)$ can be procedurally defined to drop the first element of c and append o.

To perform the sampling required for derivation tree construction we must define $P(o|c)$. Using $\mathcal{M}$, we begin by converting a large corpus of sentence segmented text into a training set of derivation trees. Maximum likelihood estimation of $P(o|c)$ is then as simple as normalizing the counts of the observed outcomes for each observed context. However, in order to obtain contexts for which the conditional independence assumption of $P(o|c)$ is appropriate, it is necessary to condition on a large amount of information. This leads to sparse estimates even on large amounts of training data, a problem that can be addressed by smoothing. We identify two complementary types of smoothing, and illustrate them with the following sentences.

> The furry dog bit me.
> The cute cat licked me.

An unsmoothed bigram model trained on this data can only generate the two sentences verbatim. If, however, we know that the tokens "dog" and "cat" are semantically similar, we can smooth by assuming the words that follow "cat" are also likely to follow "dog". This is easily handled with

traditional smoothing techniques that interpolate between distributions estimated for both coarse, $P(w|w_{-1}$=[*animal*]), and fine, $P(w|w_{-1}$="dog"), contexts. We refer to this as *context smoothing*.

However, we would also like to capture the intuition that words which can be followed by "dog" can also be followed by "cat", which we will call *outcome smoothing*. We extend our terminology to describe a system that performs both types of smoothing with the following

- the set $\bar{\mathcal{C}}$ of smooth contexts $\bar{c}$
- the set $\bar{\mathcal{O}}$ of smooth outcomes $\bar{o}$
- a smoothing function $S_C : \mathcal{C} \rightarrow \bar{\mathcal{C}}$
- a smoothing function $S_O : \mathcal{O} \rightarrow \bar{\mathcal{O}}$

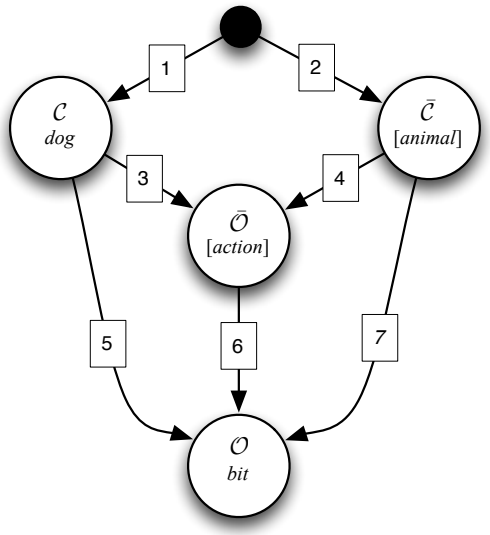

Figure 1: A flow chart depicting the decisions made when choosing an outcome for a context. The large circles show the set of items associated with each decision, and contain examples items for a bigram model where S_C and S_O map words (e.g. *dog*) to semantic classes (e.g. [*animal*]).

We describe the smoothed generative process with the flowchart shown in Figure 1. In order to choose an outcome for a given context, two decisions must be made. First, we must decide which context we will employ, the true context or the smooth context, marked by edges 1 or 2 respectively. Next, we choose to generate a true outcome or a smooth outcome, and if we select the latter we use edge 6 to choose a true outcome given the smooth outcome. The decision between edges 1 and 2 can be sampled from a Bernoulli random variable with parameter λ_c, with one variable estimated for each context c. The decision between edges 5 and 3 and the one between 4 and 7 can also be made with Bernoulli random variables, with parameter sets γ_c and $\gamma_{\bar{c}}$ respectively.

This yields the full form of the unconstrained probabilistic generative model as follows

$$P(o|c) = \lambda_c P_1(o|c) + (1 - \lambda_c) P_2(o|S_C(c))$$

$$\begin{aligned} P_1(o|c) = {} & \gamma_c P_5(o|c) + \\ & (1 - \gamma_c) P_7(o|\bar{o}) P_3(\bar{o}|c) \end{aligned} \quad (1)$$

$$\begin{aligned} P_2(o|\bar{c}) = {} & \gamma_{\bar{c}} P_6(o|c) + \\ & (1 - \gamma_{\bar{c}}) P_7(o|\bar{o}) P_4(\bar{o}|\bar{c}) \end{aligned}$$

requiring estimation of the λ and γ variables as well as the five multinomial distributions P_{3-7}. This can be done with a straightforward application of EM.

4 Limiting Vocabulary

A primary concern in the generation of language education exercises is the working vocabulary of the students. If efficiency were not a concern, the natural solution to the vocabulary constraint would be rejection sampling: simply generate sentences until one happens to obey the constraint. In this section we show how to generate a sentence directly from this constrained set with a distribution closely approximating that of the rejection sampler.

4.1 Pruning

The first step is to prune the space of possible sentences to those that obey the vocabulary constraint. For the models we investigate there is a natural predicate $V(o)$ that is true if and only if an outcome introduces a word that is out of vocab, and so the vocabulary constraint is equivalent to the requirement that $V(o)$ is false for all possible outcomes o. Considering transitions along edges in Figure 1, the removal of all transitions along edges 5,6, and 7 that lead to outcomes where $V(o)$ is true satisfies this property.

Our remaining concern is that the generation process does not reach a failure case. Again considering transitions in Figure 1, failure occurs when we require $P(o|c)$ for some c and there is no transition to c on edge 1 or $S_C(c)$ along edge 2. We refer to such a context as *invalid*. Our goal, which we refer to as *consistency*, is that for all

valid contexts c, all outcomes o that can be reached in Figure 1 satisfy the property that all members of $I(c, o)$ are valid contexts.

To see how we might end up in failure, consider a trigram model on POS/word pairs for which S_C is the identity function and S_O backs off to the POS tag. Given a context $c = (\binom{t_{-2}}{w_{-2}}, \binom{t_{-1}}{w_{-1}})$ if we generate along a path using edge 6 we will choose a smooth outcome t_0 that we have seen following c in the data and then independenently choose a w_0 that has been observed with tag t_0. This implies a following context $(\binom{t_{-1}}{w_{-1}}, \binom{t_0}{w_0})$. If we have estimated our model with observations from data, there is no guarantee that this context ever appeared, and if so there will be no available transition along edges 1 or 2.

Let the list $\bar{I}(c, o)$ be the result of the mapped application of S_C to each element of $I(c, o)$. In order to define an efficient algorithm, we require the following property $\mathbf{D}$ referring to the amount of information needed to determine $\bar{I}(c, o)$. Simply put, $\mathbf{D}$ states if the smoothed context and outcome are fixed, then the implied smooth contexts are determined.

$\mathbf{D}$ $\{S_C(c), S_O(o)\} \rightarrow \bar{I}(c, o)$

To highlight the statement $\mathbf{D}$ makes, consider the trigram POS/word model described above, but let S_C also map the POS/word pairs in the context to their POS tags alone. $\mathbf{D}$ holds here because given $S_C(c) = (t_{-2}, t_{-1})$ and $S_O(o) = t_0$ from the outcome, we are able to determine the implied smooth context (t_{-1}, t_0). If context smoothing instead produced $S_C(c) = (t_{-2})$, $\mathbf{D}$ would not hold.

If $\mathbf{D}$ holds then we can show consistency based on the transitions in Figure 1 alone as any complete path through Figure 1 defines both $\bar{c}$ and $\bar{o}$. By $\mathbf{D}$ we can determine $\bar{I}(c, o)$ for any path and verify that all its members have possible transitions along edge 2. If the verification passes for all paths then the model is consistent.

Algorithm 1 produces a consistent model by verifying each complete path in the manner just described. One important feature is that it preserves the invariant that if a context c can be reached on edge 1, then $S_C(c)$ can be reached on edge 2. This means that if the verification fails then the complete path produces an invalid context, even though we have only checked the members of $\bar{I}(c, o)$ against path 2.

If a complete path produces an invalid context, some transition along that path must be removed. It is never optimal to remove transitions from edges 1 or 2 as this unnecessarily removes all downstream complete paths as well, and so for invalid complete paths along 1-5 and 2-7 Algorithm 1 removes the transitions along edges 5 and 7. The choice is not so simple for the complete paths 1-3-6 and 2-4-6, as there are two remaining choices. Fortunately, $\mathbf{D}$ implies that breaking the connection on edge 3 or 4 is optimal as regardless of which outcome is chosen on edge 6, $\bar{I}(c, o)$ will still produce the same invalid $\bar{c}$.

Algorithm 1 Pruning Algorithm

```
Initialize with all observed transitions
for all out of vocab o do
  remove ? → o from edges 5,6, and 7
end for
repeat
  for all paths in flow chart do
    if ∃c̄ ∈ Ī(c, o) s.t. c̄ is invalid then
      remove transition from edge 5,7,3 or 4
    end if
  end for
  Run FIXUP
until edge 2 transitions did not change
```

After removing transitions in this manner, some transitions on edges 1-4 may no longer have any outgoing transitions. The subroutine FIXUP removes such transitions, checking edges 3 and 4 before 1 and 2. If FIXUP does not modify edge 2 then the model is consistent and Algorithm 1 terminates.

4.2 Estimation

In order to replicate the behavior of the rejection sampler, which uses the original probability model $P(o|c)$ from Equation 1, we must set the probabilities $P_V(o|c)$ of the pruned model appropriately. We note that for moderately sized vocabularies it is feasible to recursively enumerate $\mathcal{C}_V$, the set of all reachable contexts in the pruned model. In further discussion we simplify the representation of the model to a standard PCFG with $\mathcal{C}_V$ as its symbol set and its PCFG rules indexed by outcomes. This also allows us to construct the *reachability graph* for $\mathcal{C}_V$, with an edge from c_i to c_j for each $c_j \in I(c_i, o)$. Such an edge is given weight $P(o|c)$, the probability under the unconstrained model, and zero weight edges are not included.

Our goal is to retain the form of the stan-

dard incremental recursive sampling algorithm for PCFGs. The correctness of this algorithm comes from the fact that the probability of a rule R expanding a symbol X is precisely the probability of all trees rooted at X whose first rule is R. This implies that the correct sampling distribution is simply the distribution over rules itself. When constraints that disallow certain trees are introduced, the probability of all trees whose first rule is R only includes the mass from valid trees, and the correct sampling distribution is the renormalization of these values.

Let the *goodness* of a context $G(c)$ be the probability that a full subtree generated from c using the unconstrained model obeys the vocabulary constraint. Knowledge of $G(c)$ for all $c \in \mathcal{C}_V$ allows the calculation of probabilities for the pruned model with

$$P_V(o|c) \propto P(o|c) \prod_{c' \in I(c,o)} G(c') \qquad (2)$$

While $G(c)$ can be defined recursively as

$$G(c) = \sum_{o \in \mathcal{O}} P(o|c) \prod_{c' \in I(c,o)} G(c') \qquad (3)$$

its calculation requires that the reachability graph be acyclic. We approximate an acyclic graph by listing all edges in order of decreasing weight and introducing edges as long as they do not create cycles. This can be done efficiently with a binary search over the edges by weight. Note that this approximate graph is used only in recursive estimation of $G(c)$, and the true graph can still be used in Equation 2.

5 Generating Up

In this section we show how to efficiently generate sentences that contain an arbitrary word w^* in addition to the vocabulary constraint. We assume the ability to easily find $\mathcal{C}_{w^*}$, a subset of $\mathcal{C}_V$ whose use guarantees that the resulting sentence contains w^*. Our goal is once again to efficiently emulate the rejection sampler, which generates a derivation tree T and accepts if and only if it contains at least one member of $\mathcal{C}_{w^*}$.

Let $\mathcal{T}_{w^*}$ be the set of derivation trees that would be accepted by the rejection sampler. We present a three stage generative model and its associated probability distribution $P_{w^*}(\tau)$ over items τ for which there is a functional mapping into $\mathcal{T}_{w^*}$.

In addition to the probabilities $P_V(o|c)$ from the previous section, we require an estimate of $\mathbb{E}(c)$, the expected number of times each context c appears in a single tree. This can be computed efficiently using the mean matrix, described in Miller and Osullivan (1992). This $|\mathcal{C}_V| \times |\mathcal{C}_V|$ matri x M has its entries defined as

$$M(i,j) = \sum_{o \in \mathcal{O}} P(o|c_i) \#(c_j, c_i, o) \qquad (4)$$

where the operator $\#$ returns the number of times context c_j appears $I(c_i, o)$. Defining a $1 \times |\mathcal{C}_V|$ start state vector z_0 that is zero everywhere and 1 in the entry corresponding to the root context gives

$$\mathbb{E}(z) = \sum_{i=0}^{\infty} z_0 M^i$$

which can be iteratively computed with sparse matrix multiplication. Note that the ith term in the sum corresponds to expected counts at depth i in the derivation tree. With definitions of context and outcome for which very deep derivations are improbable, it is reasonable to approximate this sum by truncation.

Our generation model operates in three phases.

1. Chose a start context $c_0 \in \mathcal{C}_{w^*}$
2. Generate a spine S of contexts and outcomes connecting c_0 to the root context
3. Fill in the full derivation tree T below all remaining unexpanded contexts

In the first phase, c_0 is sampled from the multinomial

$$P_1(c_0) = \frac{\mathbb{E}(c_0)}{\sum_{c \in \mathcal{C}_{w^*}} \mathbb{E}(c)} \qquad (5)$$

The second step produces a spine S, which is formally an ordered list of triples. Each element of S records a context c_i, an outcome o_i, and the index k in $I(c_i, o_i)$ of the child along which the spine progresses. The members of S are sampled independantly given the previously sampled context, starting from c_0 and terminating when the root context is reached. Intuitively this is equivalent to generating the path from the root to c_0 in a bottom up fashion.

We define the probability P_σ of a triple (c_i, o_i, k) given a previously sampled context c_j

as

$$P_\sigma(\{c_i, o_i, k\}|c_j) \propto \begin{cases} \mathbb{E}(c_i)P_V(o_i|c_i) & I(c_i, o_i)[k] = c_j \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

Let $S = (c_1, o_1, k_1) \dots (c_n, o_n, k_n)$ be the results of this recursive sampling algorithm, where c_n is the root context, and c_1 is the parent context of c_0. The total probability of a spine S is then

$$P_2(S|c_0) = \prod_{i=1}^{|S|} \frac{\mathbb{E}(c_i)P_V(o_i|c_i)}{Z_{i-1}} \quad (7)$$

$$Z_{i-1} = \sum_{(c,o)\in\mathbb{I}_{c_{i-1}}} \mathbb{E}(c)P_V(o|c)\#(c_{i-1}, c, o) \quad (8)$$

where $\mathbb{I}_{c-1}$ is the set of all (c, o) for which $P_\sigma(c, o, k|c_{i-1})$ is non-zero for some k. A key observation is that $Z_{i-1} = \mathbb{E}(c_{i-1})$, which cancels nearly all of the expected counts from the full product. Along with the fact that the expected count of the root context is one, the formula simplifies to

$$P_2(S|c_0) = \frac{\prod_{i=1}^{|S|} P_V(o_i|c_i)}{\mathbb{E}(c_0)} \quad (9)$$

The third step generates a final tree T by filling in subtrees below unexpanded contexts on the spine S using the original generation algorithm, yielding results with probability

$$P_3(T|S) = \prod_{(c,o)\in T/S} P_V(o|c) \quad (10)$$

where the set T/S includes all contexts that are not ancestors of c_0, as their outcomes are already specified in S.

We validate this algorithm by considering its distrubution over complete derivation trees $T \in \mathcal{T}_{w^*}$. The algorithm generates $\tau = (T, S, c_0)$ and has a simple functional mapping into $\mathcal{T}_{w^*}$ by extracting the first member of τ.

Combining the probabilities of our three steps gives

$$P_{w^*}(\tau) = \frac{\mathbb{E}(c_0)}{\sum_{c\in\mathcal{C}_{w^*}} \mathbb{E}(c)} \frac{\prod_{i=1}^{|S|} P_V(o_i|c_i)}{\mathbb{E}(c_0)} \prod_{(c,o)\in T/S} P_V(o|c)$$

$$P_{w^*}(\tau) = \frac{P_V(T)}{\sum_{c\in\mathcal{C}_{w^*}} \mathbb{E}(c)} = \frac{1}{\rho} P_V(T) \quad (11)$$

where ρ is a constant and

$$P_V(T) = \prod_{(c,o)\in T} P_V(o|c)$$

is the probability of T under the original model. Note that several τ may map to the same T by using different spines, and so

$$P_{w^*}(T) = \frac{\eta(T)}{\rho} P_V(T) \quad (12)$$

where $\eta(T)$ is the number of possible spines, or equivalently the number of contexts $c \in \mathcal{C}_{w^*}$ in T.

Recall that our goal is to efficiently emulate the output of a rejection sampler. An ideal system P_{w^*} would produce the complete set of derivation trees accepted by the rejection sampler using P_V, with probabilities of each derivation tree T satisfying

$$P_{w^*}(T) \propto P_V(T) \quad (13)$$

Consider the implications of the following assumption

A each $T \in \mathcal{T}_{w^*}$ contains exactly one $c \in \mathcal{C}_{w^*}$

A ensures that $\eta(T) = 1$ for all T, unifying Equations 12 and 13. **A** does not generally hold in practice, but its clear exposition allows us to design models for which it holds most of the time, leading to a tight approximation.

The most important consideration of this type is to limit redundancy in $\mathcal{C}_{w^*}$. For illustration consider a dependency grammar model with parent annotation where a context is the current word and its parent word. When specifying $\mathcal{C}_{w^*}$ for a particular w^*, we might choose all contexts in which w^* appears as either the current or parent word, but a better choice that more closely satisfies **A** is to choose contexts where w^* appears as the current word only.

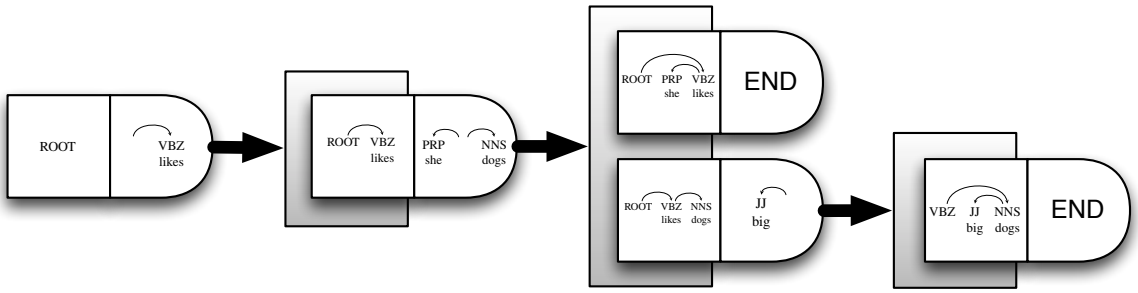

Figure 2: The generation system SPINEDEP draws on dependency tree syntax where we use the term *node* to refer to a POS/word pair. Contexts consist of a node, its parent node, and grandparent POS tag, as shown in squares. Outcomes, shown in squares with rounded right sides, are full lists of dependents or the END symbol. The shaded rectangles contain the results of $I(c, o)$ from the indicated (c, o) pair.

6 Experiments

We train our models on sentences drawn from the Simple English Wikipedia[1]. We obtained these sentences from a data dump which we liberally filtered to remove items such as lists and sentences longer than 15 words or shorter then 3 words. We parsed this data with the recently updated Stanford Parser (Socher et al., 2013) to Penn Treebank constituent form, and removed any sentence that did not parse to a top level S containing at least one NP and one VP child. Even with such strong filters, we retained over 140K sentences for use as training data, and provide this exact set of parse trees for use in future work.[2]

Inspired by the application in language education, for our vocabulary list we use the English Vocabulary Profile (Capel, 2012), which predicts student vocabulary at different stages of learning English as a second language. We take the most basic American English vocabulary (the A1 list), and retrieve all inflections for each word using SimpleNLG (Gatt and Reiter, 2009), yielding a vocabulary of 1226 simple words and punctuation.

To mitigate noise in the data, we discard any pair of context and outcome that appears only once in the training data, and estimate the parameters of the unconstrained model using EM.

6.1 Model Comparison

We experimented with many generation models before converging on SPINEDEP, described in Figure 2, which we use in these experiments.

[1]http://simple.wikipedia.org

[2]http://bllip.cs.brown.edu/download/simplewiki.ptb

	Corr(%)	% uniq
SPINEDEP unsmoothed	87.6	5.0
SPINEDEP WordNet	78.3	32.5
SPINEDEP word2vec 5000	72.6	52.9
SPINEDEP word2vec 500	65.3	60.2
KneserNey-5	64.0	25.8
DMV	33.7	71.2

Figure 3: System comparison based on human judged correctness and the percentage of unique sentences in a sample of 100K.

SPINEDEP uses dependency grammar elements, with parent and grandparent information in the contexts to capture such distinctions as that between main and clausal verbs. Its outcomes are full configurations of dependents, capturing coordinations such as subject-object pairings. This specificity greatly increases the size of the model and in turn reduces the speed of the true rejection sampler, which fails over 90% of the time to produce an in-vocab sentence.

We found that large amounts of smoothing quickly diminishes the amount of error free output, and so we smooth very cautiously, mapping words in the contexts and outcomes to fine semantic classes. We compare the use of human annotated hypernyms from Wordnet (Miller, 1995) with automatic word clusters from word2vec (Mikolov et al., 2013), based on vector space word embeddings, evaluating both 500 and 5000 clusters for the latter.

We compare these models against several baseline alternatives, shown in Figure 3. To determine

correctness, used Amazon Mechanical Turk, asking the question: "Is this sentence plausible?". We further clarified this question in the instructions with alternative definitions of plausibility as well as both positive and negative examples. Every sentence was rated by five reviewers and its correctness was determined by majority vote, with a .496 Fleiss kappa agreement. To avoid spammers, we limited our hits to Turkers with an over 95% approval rating.

Traditional language modeling techniques such as such as the Dependency Model with Valence (Klein and Manning, 2004) and 5-gram Kneser Ney (Chen and Goodman, 1996) perform poorly, which is unsurprising as they are designed for tasks in recognition rather than generation. For n-gram models, accuracy can be greatly increased by decreasing the amount of smoothing, but it becomes difficult to find long n-grams that are completely in-vocab and results become redundant, parroting the few completely in-vocab sentences from the training data. The DMV is more flexible, but makes assumptions of conditional independence that are far too strong. As a result it is unable to avoid red flags such as sentences not ending in punctuation or strange subject-object coordinations. Without smoothing, SPINEDEP suffers from a similar problem as unsmoothed n-gram models; high accuracy but quickly vanishing productivity.

All of the smoothed SPINEDEP systems show clear advantages over their competitors. The tradeoff between correctness and generative capacity is also clear, and our results suggest that the number of clusters created from the word2vec embeddings can be used to trace this curve. As for the ideal position in this tradeoff, we leave such decisions which are particular to specific application to future work, arbitrarily using SPINEDEP WordNet for our following experiments.

6.2 Fixed Vocabulary

To show the tightness of the approximation presented in Section 4.2, we evaluate three settings for the probabilities of the pruned model. The first is a weak baseline that sets all distributions to uniform. For the second, we simply renormalize the true model's probabilities, which is equivalent to setting $G(c) = 1$ for all c in Equation 2. Finally, we use our proposed method to estimate $G(c)$.

We show in Figure 4 that our estimation method more closely approximates the distribution of the rejection sampler by drawing 500K samples from each model and comparing them with 500K samples from the rejection sampler itself. We quantify this comparison with the likelihood ratio statistic, evaluating the null hypothesis that the two samples were drawn from the same distribution. Not only does our method more closely emulate that of the rejection sampler, be we see welcome evidence that closeness to the true distribution is correlated with correctness.

	Corr(%)	-LLR
True RS	79.3	–
Uniform	47.3	96.2
$G(c)$ = 1	77.0	25.0
$G(c)$ estimated	**78.3**	**1.0**

Figure 4: A comparison of our system against both a weak and a strong baseline based on correctness and the negative log of the likelihood ratio measuring closeness to the true rejection sampler.

6.3 Word Inclusion

To explore the word inclusion constraint, for each word in our vocabulary list we sample 1000 sentences that are constrained to include that word using both unsmoothed and WordNet smoothed SPINEDEP. We compare these results to the "Corpus" model that simply searches the training data and uniformly samples from the existing sentences that satisfy the constraints. This corpus search approach is quite a strong baseline, as it is trivial to implement and we assume perfect correctness for its results.

This experiment is especially relevant to our motivation of language education. The natural question when proposing any NLG approach is whether or not the ability to automatically produce sentences outweighs the requirement of a post-process to ensure goal-appropriate output. This is a challenging task in the context of language education, as most applications such as exam or homework creation require only a handful of sentences. In order for an NLG solution to be appropriate, the constraints must be so strong that a corpus search based method will frequently produce too few options to be useful. The word inclusion constraint highlights the strengths of our method as it is not only highly plausible in a language ed-

	# < 10	# > 100	Corr(%)
Corpus	987	26	100
Unsmooth	957	56	89.0
Smooth	544	586	79.0

Figure 5: Using systems that implement the word inclusion constraint, this table shows the number of words for which the amount of unique sentences out of 1000 samples was less than 10 or greater than 100, along with the correctness of each system.

ucation setting but difficult to satisfy by chance in large corpora.

Figure 5 shows that the corpus search approach fails to find more than ten sentences that obey the word inclusion constraints for most target words. Moreover, it is arguably the case that unsmoothed SPINEDEP is even worse due to its inferior correctness. With the addition of smoothing, however, we see a drastic shift in the number of words for which a large number of sentences can be produced. For the majority of the vocabulary words this model generates over 100 sentences that obey both constraints, of which approximately 80% are valid English sentences.

7 Conclusion

In this work we address two novel NLG constraints, fixed vocabulary and fixed vocabulary with word inclusion, that are motivated by language education scenarios. We showed that under these constraints a highly parameterized model based on dependency tree syntax can produce a wide range of accurate sentences, outperforming the strong baselines of popular generative language models. We developed a pruning and estimation algorithm for the fixed vocabulary constraint and showed that it not only closely approximates the true rejection sampler but also that the tightness of approximation is correlated with human judgments of correctness. We showed that under the word inclusion constraint, precise semantic smoothing produces a system whose abilities exceed the simple but powerful alternative of looking up sentences in large corpora.

SPINEDEP works surprisingly well given the widely held stigma that freeform NLG produces either memorized sentences or gibberish. Still, we expect that better models exist, especially in terms of definition of smoothing operators. We have presented our algorithms in the flexible terms of context and outcome, and clearly stated the properties that are required for the full use of our methodology. We have also implemented our code in these general terms[3], which performs EM based parameter estimation as well as efficient generation under the constraints discussed above. All systems used in this work with the exception of 5-gram interpolated Kneser-Ney were implemented in this way, are included with the code, and can be used as templates.

We recognize several avenues for continued work on this topic. The use of form-based constraints such as word inclusion has clear application in language education, but many other constraints are also desirable. The clearest is perhaps the ability to constrain results based on a "vocabulary" of syntactic patterns such as "Not only ... but also ...". Another extension would be to incorporate the rough communicative goal of response to a previous sentence as in Wu et al. (2013) and attempt to produce in-vocab dialogs such as are ubiquitous in language education textbooks.

Another possible direction is in the improvement of the context-outcome framework itself. While we have assumed a data set of one derivation tree per sentence, our current methods easily extend to sets of weighted derivations for each sentence. This suggests the use of techinques that have proved effective in grammar estimation that reason over large numbers of possible derivations such as Bayesian tree substitution grammars or unsupervised symbol refinement.

References

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

A. Capel. 2012. The english vocabulary profile. *http://vocabulary.englishprofile.org/*.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, ACL '96, pages 310–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Simon Colton, Jacob Goodwin, and Tony Veale. 2012. Full face poetry generation. In *Proceedings of the*

[3]https://github.com/chonger/Babble

Third International Conference on Computational Creativity, pages 95–102.

Albert Gatt and Ehud Reiter. 2009. Simplenlg: A realisation engine for practical applications. In *Proceedings of the 12th European Workshop on Natural Language Generation*, ENLG '09, pages 90–93, Stroudsburg, PA, USA. Association for Computational Linguistics.

Erica Greene, Tugba Bodrumlu, and Kevin Knight. 2010. Automatic analysis of rhythmic poetry with applications to generation and translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 524–533, Stroudsburg, PA, USA. Association for Computational Linguistics.

Long Jiang and Ming Zhou. 2008. Generating chinese couplets using a statistical mt approach. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 377–384. Association for Computational Linguistics.

Dan Klein and Christopher D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Michael I. Miller and Joseph A. Osullivan. 1992. Entropies and combinatorics of random branching processes and context-free languages. *IEEE Transactions on Information Theory*, 38.

George A. Miller. 1995. Wordnet: A lexical database for english. *COMMUNICATIONS OF THE ACM*, 38:39–41.

Margaret Mitchell, Kees van Deemter, and Ehud Reiter. 2013. Generating expressions that refer to visible objects. In *HLT-NAACL*, pages 1174–1184.

Jack Mostow and Hyeju Jang. 2012. Generating diagnostic multiple choice comprehension cloze questions. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 136–146. Association for Computational Linguistics.

Gözde Özbal, Daniele Pighin, and Carlo Strapparava. 2013. Brainsup: Brainstorming support for creative sentence generation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1446–1455, Sofia, Bulgaria, August. Association for Computational Linguistics.

Ananth Ramakrishnan A, Sankar Kuppan, and Sobha Lalitha Devi. 2009. Automatic generation of tamil lyrics for melodies. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, pages 40–46. Association for Computational Linguistics.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013. Parsing With Compositional Vector Grammars. In *ACL*.

Eiichiro Sumita, Fumiaki Sugaya, and Seiichi Yamamoto. 2005. Measuring non-native speakers' proficiency of english by using a test with automatically-generated fill-in-the-blank questions. In *Proceedings of the second workshop on Building Educational Applications Using NLP*, pages 61–68. Association for Computational Linguistics.

Alessandro Valitutti, Hannu Toivonen, Antoine Doucet, and Jukka M. Toivanen. 2013. "let everything turn well in your wife": Generation of adult humor using lexical constraints. In *ACL (2)*, pages 243–248.

Dekai Wu, Karteek Addanki, Markus Saers, and Meriem Beloucif. 2013. Learning to freestyle: Hip hop challenge-response induction via transduction rule segmentation. In *EMNLP*, pages 102–112.