Stochastic Beams and Where to Find Them:
The Gumbel-Top-k Trick for Sampling Sequences Without Replacement

Wouter Kool [1 2]   **Herke van Hoof** [1]   **Max Welling** [1 3]

Abstract

The well-known Gumbel-Max trick for sampling from a categorical distribution can be extended to sample k elements without replacement. We show how to implicitly apply this '*Gumbel-Top-k*' trick on a factorized distribution over sequences, allowing to draw exact samples without replacement using a *Stochastic Beam Search*. Even for exponentially large domains, the number of model evaluations grows only linear in k and the maximum sampled sequence length. The algorithm creates a theoretical connection between sampling and (deterministic) beam search and can be used as a principled intermediate alternative. In a translation task, the proposed method compares favourably against alternatives to obtain diverse yet good quality translations. We show that sequences sampled without replacement can be used to construct low-variance estimators for expected sentence-level BLEU score and model entropy.

1. Introduction

We think the Gumbel-Max trick (Gumbel, 1954; Maddison et al., 2014) is like a *magic* trick. It allows sampling from the categorical distribution, simply by *perturbing* the log-probability for each category by adding *independent* Gumbel distributed noise and returning the category with maximum *perturbed log-probability*. This trick has recently (re-)gained popularity as it allows to derive reparameterizable continous relaxations of the categorical distribution (Maddison et al., 2016; Jang et al., 2016). However, there is more: as was noted (in a blog post) by Vieira (2014), taking the top k largest perturbed log-probabilities (instead of the maximum, or *top 1*) yields a sample of size k from the categorical distribution *without replacement*. We refer to this

extension of the Gumbel-Max trick as the *Gumbel-Top-k* trick.

In this paper, we consider *factorized distributions over sequences*, represented by (parametric) *sequence models*. Sequence models are widely used, e.g. in tasks such as neural machine translation (Sutskever et al., 2014; Bahdanau et al., 2015) and image captioning (Vinyals et al., 2015b). Many such tasks require obtaining a set of representative sequences from the model. These can be random samples, but for low-entropy distributions, a set of sequences sampled using standard sampling (with replacement) may contain many duplicates. On the other hand, a *beam search* can find a set of unique high-probability sequences, but these have low variability and, being deterministic, cannot be used to construct statistical estimators.

In this paper, we propose sampling without replacement as an alternative method to obtain representative sequences from a sequence model. We show that for a sequence model, we can draw samples without replacement by applying the Gumbel-Top-k trick *implicitly*, without instantiating all sequences in the (typically exponentially large) domain. This procedure allows us to draw unique sequences using a number of model evaluations that grows only *linear* in the number of samples k and the maximum sampled sequence length. The algorithm uses the idea of top-down sampling (Maddison et al., 2014) and performs a beam search over stochastically perturbed log-probabilities, which is why we refer to it as *Stochastic Beam Search*.

Stochastic Beam Search is a novel procedure for sampling sequences that avoids duplicate samples. Unlike ordinary beam search, *it has a probabilistic interpretation.* Thus, it can e.g. be used in importance sampling. As such, Stochastic Beam Search conceptually connects sampling and beam search, and combines advantages of both methods. In Section 4 we give two examples of how Stochastic Beam Search can be used as a principled alternative to sampling or beam search. In these experiments, Stochastic Beam Search is used to control the diversity of translation results, as well as to construct low variance estimators for sentence-level BLEU score and model entropy.

---

[1]University of Amsterdam, The Netherlands [2]ORTEC, The Netherlands [3]CIFAR, Canada. Correspondence to: Wouter Kool <w.w.m.kool@uva.nl>.

2. Preliminaries

2.1. The categorical distribution

A discrete random variable I has a distribution Categorical$(p_1, ..., p_n)$ with domain $N = \{1, ..., n\}$ if $P(I = i) = p_i \quad \forall i \in N$. We refer to the categories i as *elements* in the domain N and denote with $\phi_i, i \in N$ the *(unnormalized) log-probabilities*, so $\exp \phi_i \propto p_i$ and $p_i = \frac{\exp \phi_i}{\sum_{j \in N} \exp \phi_j}$. Therefore in general we write:

$$I \sim \text{Categorical} \left(\frac{\exp \phi_i}{\sum_{j \in N} \exp \phi_j}, i \in N \right). \quad (1)$$

2.2. The Gumbel distribution

If $U \sim \text{Uniform}(0, 1)$, then $G = \phi - \log(-\log U)$ has a *Gumbel* distribution with location ϕ and we write $G \sim \text{Gumbel}(\phi)$. From this it follows that $G' = G + \phi' \sim \text{Gumbel}(\phi + \phi')$, so we can *shift* Gumbel variables.

2.3. The Gumbel-Max trick

The Gumbel-Max trick (Gumbel, 1954; Maddison et al., 2014) allows to sample from the categorical distribution (1) by independently *perturbing* the log-probabilities ϕ_i with Gumbel noise and finding the largest element.

Formally, let $G_i \sim \text{Gumbel}(0), i \in N$ i.i.d. and let $I^* = \arg \max_i \{\phi_i + G_i\}$, then $I^* \sim \text{Categorical}(p_i, i \in N)$ with $p_i \propto \exp \phi_i$. In a slight abuse of notation, we write $G_{\phi_i} = G_i + \phi_i \sim \text{Gumbel}(\phi_i)$ and we call G_{ϕ_i} the *perturbed log-probability* of element or category i in the domain N.

For any subset $B \subseteq N$ it holds that (Maddison et al., 2014):

$$\max_{i \in B} G_{\phi_i} \sim \text{Gumbel} \left(\log \sum_{j \in B} \exp \phi_j \right), \quad (2)$$

$$\arg \max_{i \in B} G_{\phi_i} \sim \text{Categorical} \left(\frac{\exp \phi_i}{\sum_{j \in B} \exp \phi_j}, i \in B \right). \quad (3)$$

Additionally, the max (2) and argmax (3) are independent. For details, see Maddison et al. (2014).

2.4. The Gumbel-Top-k trick

Considering the maximum the *top 1* (one), we can generalize the Gumbel-Max trick to the Gumbel-Top-k trick to draw an *ordered sample of size k without replacement*[1], by

---

[1] With sampling without replacement from a categorical distribution, we mean sampling the first element, then renormalizing the remaining probabilities to sample the next element, etcetera. This does *not* mean that the *inclusion probability* of element i is proportional to p_i: if we sample $k = n$ elements all elements are included with probability 1.

taking the indices of the k largest perturbed log-probabilities. Generalizing $\arg \max$, we denote with $\arg \text{top} \, k$ the function that takes a sequence of values and returns the indices of the k largest values, in order of decreasing value.

Theorem 1. *For $k \leq n$, let $I_1^*, ..., I_k^* = \arg \text{top} \, k \, G_{\phi_i}$. Then $I_1^*, ..., I_k^*$ is an (ordered) sample without replacement from the Categorical $\left(\frac{\exp \phi_i}{\sum_{j \in N} \exp \phi_j}, i \in N \right)$ distribution, e.g. for a realization $i_1^*, ..., i_k^*$ it holds that*

$$P\left(I_1^* = i_1^*, ..., I_k^* = i_k^*\right) = \prod_{j=1}^{k} \frac{\exp \phi_{i_j^*}}{\sum_{\ell \in N_j^*} \exp \phi_\ell} \quad (4)$$

where $N_j^* = N \setminus \{i_1^*, ..., i_{j-1}^*\}$ is the domain (without replacement) for the j-th sampled element.

For more info we refer to the blog post by Vieira (2014). We include a proof in Appendix A.

2.5. Sequence models

A *sequence model* is a factorized parametric distribution over sequences. The parameters $\boldsymbol{\theta}$ define the conditional probability $p_{\boldsymbol{\theta}}(y_t | \boldsymbol{y}_{1:t-1})$ of the next token y_t given the partial sequence $\boldsymbol{y}_{1:t-1}$. Typically $p_{\boldsymbol{\theta}}(y_t | \boldsymbol{y}_{1:t-1})$ is defined as a softmax normalization of unnormalized log-probabilities $\phi_{\boldsymbol{\theta}}(y_t | \boldsymbol{y}_{1:t-1})$ with optional temperature T (default $T = 1$):

$$p_{\boldsymbol{\theta}}(y_t | \boldsymbol{y}_{1:l-1}) = \frac{\exp \left(\phi_{\boldsymbol{\theta}}(y_t | \boldsymbol{y}_{1:t-1}) / T \right)}{\sum_{y'} \exp \left(\phi_{\boldsymbol{\theta}}(y' | \boldsymbol{y}_{1:t-1}) / T \right)}. \quad (5)$$

The normalization is w.r.t. a single token, so the model is *locally normalized*. The total probability of a (partial) sequence $\boldsymbol{y}_{1:l}$ follows from the chain rule of probability:

$$p_{\boldsymbol{\theta}}(\boldsymbol{y}_{1:t}) = p_{\boldsymbol{\theta}}(y_t | \boldsymbol{y}_{1:t-1}) \cdot p_{\boldsymbol{\theta}}(\boldsymbol{y}_{1:t-1}) \quad (6)$$

$$= \prod_{t'=1}^{t} p_{\boldsymbol{\theta}}(y_{t'} | \boldsymbol{y}_{1:t'-1}). \quad (7)$$

A sequence model defines a valid probability distribution over both partial and complete sequences. When the length is irrelevant, we simply write $\boldsymbol{y}$ to indicate a (partial or complete) sequence. If the model is additionally conditioned on a *context* $\boldsymbol{x}$ (e.g., a source sentence), we write $p_{\boldsymbol{\theta}}(\boldsymbol{y} | \boldsymbol{x})$.

2.6. Beam Search

A *beam search* is a limited-width breadth first search. In the context of sequence models, it is often used as an approximation to finding the (single) sequence $\boldsymbol{y}$ that maximizes (7), or as a way to obtain a set of high-probability sequences from the model. Starting from an empty sequence (e.g. $t = 0$), a beam search expands at every step $t = 0, 1, 2, ...$ at most k partial sequences (those with highest probability) to compute the probabilities of sequences with length $t + 1$. It terminates with a beam of k complete sequences, which we assume to be of equal length (as they can be padded).

Figure 1. Example of the Gumbel-Top-k trick on a tree, with $k = 3$. The bars next to the leaves indicate the perturbed log-probabilities G_{ϕ_i}, while the bars next to internal nodes indicate the maximum perturbed log-probability of the set of leaves S in the subtree rooted at that node: $G_{\phi_S} = \max_{i \in S} G_{\phi_i} \sim \text{Gumbel}(\phi_S)$ with $\phi_S = \log \sum_{i \in S} \exp \phi_i$. The bar is split in two to illustrate that $G_{\phi_S} = \phi_S + G_S$. Numbers in the nodes represent $p_{\boldsymbol{\theta}}(\boldsymbol{y}^S) = \exp \phi_S = \sum_{i \in S} \exp \phi_i$, the probability of the partial sequence $\boldsymbol{y}^S$. Numbers at edges represent the conditional probabilities for the next token. The shaded nodes are ancestors of the top k leaves with highest perturbed log-probability G_{ϕ_i}. These are the ones we actually need to expand. In each layer, there are at most k such nodes, such that we are guaranteed to construct all top k leaves by expanding at least the top k nodes (ranked on G_{ϕ_S}) in each level (indicated by a solid border).

3. Stochastic Beam Search

We derive Stochastic Beam Search by starting with the *explicit* application of the Gumbel-Top-k trick to sample k sequences without replacement from a sequence model. This requires instantiating all sequences in the domain to find the k largest perturbed log-probabilities. Then we transition to *top-down* sampling of the perturbed log-probabilities, and we use Stochastic Beam Search to instantiate (only) the sequences with the k largest perturbed log-probabilities. As both methods are equivalent, Stochastic Beam Search *implicitly* applies the Gumbel-Top-k trick and thus yields a sample of k sequences without replacement.

3.1. The Gumbel-Top-k trick on a tree

We represent the sequence model (7) as a tree (as in Figure 1), where internal nodes at level t represent partial sequences $\boldsymbol{y}_{1:t}$, and leaf nodes represent completed sequences. We identify a leaf by its index $i \in N = \{1, ..., n\}$ and write $\boldsymbol{y}^i$ as the corresponding sequence, with (normalized!) log-probability $\phi_i = \log p_{\boldsymbol{\theta}}(\boldsymbol{y}^i)$. To obtain a sample from the distribution (7) without replacement, we should sample from the set of leaf nodes N without replacement, for which we can naively use the Gumbel-Top-k trick (Section 2.3):

- Compute $\phi_i = \log p_{\boldsymbol{\theta}}(\boldsymbol{y}^i)$ for *all sequences* $\boldsymbol{y}^i, i \in N$. To reuse computations for partial sequences, the complete probability tree is instantiated, as in Figure 1.
- Sample $G_{\phi_i} \sim \text{Gumbel}(\phi_i)$, so G_{ϕ_i} can be seen as the perturbed log-probability of sequence $\boldsymbol{y}^i$.
- Let $i_1^*, ..., i_k^* = \arg\operatorname{top} k \, G_{\phi_i}$, then $\boldsymbol{y}^{i_1^*}, ..., \boldsymbol{y}^{i_k^*}$ is a sample of sequences from (7) without replacement.

As instantiating the complete probability tree is computationally prohibitive, we construct an equivalent process that only requires computation linear in the number of samples k and the sequence length.

Perturbed log-probabilities of partial sequences. For the naive implementation of the Gumbel-Top-k trick, we only defined the perturbed log-probabilities G_{ϕ_i} for leaf nodes i, which correspond to *complete sequences* $\boldsymbol{y}^i$. For the Stochastic Beam Search implementation, we also define the perturbed log-probabilities for *internal nodes* corresponding to *partial sequences*. We identify a node (internal or leaf) by the set S of leaves in the corresponding subtree, and we write $\boldsymbol{y}^S$ as the corresponding (partial or completed) sequence. Its log-probability $\phi_S = \log p_{\boldsymbol{\theta}}(\boldsymbol{y}^S)$ can be computed incrementally from the parent log-probability using (6), and since the model is locally normalized, it holds that

$$\phi_S = \log p_{\boldsymbol{\theta}}(\boldsymbol{y}^S) = \log \sum_{i \in S} \exp \phi_i. \qquad (8)$$

Now for each node S, we define G_{ϕ_S} as the *maximum of the perturbed log-probabilities G_{ϕ_i} in the subtree leaves S*. By Equation (2), G_{ϕ_S} has a Gumbel distribution with location ϕ_S (hence its notation G_{ϕ_S}):

$$G_{\phi_S} = \max_{i \in S} G_{\phi_i} \sim \text{Gumbel}(\phi_S) \qquad (9)$$

Since $G_{\phi_S} \sim \text{Gumbel}(\phi_S)$ is a Gumbel perturbation of the log-probability $\phi_S = \log p_{\boldsymbol{\theta}}(\boldsymbol{y}^S)$, we call it the *perturbed log-probability* of the partial sequence $\boldsymbol{y}^S$. We can define the corresponding Gumbel noise $G_S \sim \text{Gumbel}(0)$, which can be inferred from G_{ϕ_S} by the relation $G_{\phi_S} = \phi_S + G_S$.

Bottom-up sampling of perturbed log-probabilities.
We can recursively compute (9). Write Children(S) as the as the set of direct children of the node S (so Children(S) is a partition of the set S). Since the maximum (9) must be attained in one of the subtrees, it holds that

$$G_{\phi_S} = \max_{S' \in \text{Children}(S)} G_{\phi_{S'}}. \tag{10}$$

If we want to sample G_{ϕ_S} for all nodes, we can use the *bottom-up* sampling procedure: sample the leaves $G_{\phi_{\{i\}}} = G_{\phi_i}, i \in N$ and recursively compute G_{ϕ_S} using (10). This is effectively sampling from the degenerate (constant) distribution resulting from conditioning on the children.

Top-down sampling of perturbed log-probabilities.
The recursive bottom-up sampling procedure can be interpreted as ancestral sampling from a tree-structured graphical model (somewhat like Figure 1) with edges directed upwards. Alternatively, we can reverse the graphical model and sample the tree *top-down*, starting with the root and recursively sampling the children conditionally.

Note that for the root N (since it contains all leaves N), it holds that $\phi_N = \log \sum_{i \in N} \exp \phi_i = 0$, so we can let $G_{\phi_N} \sim \text{Gumbel}(0)$[2]. Starting with $S = N$, we can recursively sample the children conditionally on the parent variable G_{ϕ_S}. For $S' \in \text{Children}(S)$ it holds that $\phi_{S'} = \log p_{\boldsymbol{\theta}}(\boldsymbol{y}^{S'})$ and we can sample $G_{\phi_{S'}} \sim \text{Gumbel}(\phi_{S'})$ *conditionally* on (10), e.g. with *their maximum equal to G_{ϕ_S}*.

Sampling a set of Gumbels conditionally on their maximum being equal to a certain value is non-trivial, but can be done by first sampling the $\arg\max$ and then sampling the individual Gumbels conditionally on both the $\max$ and $\arg\max$. Alternatively, we can let $G_{\phi_{S'}} \sim \text{Gumbel}(\phi_{S'})$ independently and let $Z = \max_{S' \in \text{Children}(S)} G_{\phi_{S'}}$. Then

$$\tilde{G}_{\phi_{S'}} = -\log(\exp(-G_{\phi_S}) - \exp(-Z) + \exp(-G_{\phi_{S'}}))$$

is a set of Gumbels with a maximum equal to G_{ϕ_S}. See Appendix B for details and numerically stable implementation.

If we recursively sample the complete tree top-down, this is equivalent to sampling the complete tree bottom-up, and as a result, for all leaves, it holds that $(G_{\phi_{\{i\}}} =) G_{\phi_i} \sim \text{Gumbel}(\phi_i)$, *independently*. The benefit of using top-down sampling is that if we are interested only in obtaining the top k leaves, we do *not* need to instantiate the complete tree.

Stochastic Beam Search The key idea of Stochastic Beam Search is to apply the Gumbel-Top-k trick for a sequence model, without instantiating the entire tree, by using top-down sampling. With top-down sampling, to find the top

---

[2]Or we can simply set (e.g. condition on) $G_{\phi_N} = 0$. This does not affect the result by the independence of $\max$ and $\arg\max$.

k leaves, at every level in the tree we can suffice with only *expanding* (instantiating the subtree for) the k nodes with highest perturbed log-probability G_{ϕ_S}. To see this, first assume that we instantiated the complete tree using top-down sampling and consider the nodes that are ancestors of at least one of the top k leaves (the shaded nodes in Figure 1). At every level t of the tree, there will be at most k such nodes (as each of the top k leaves has only one ancestor at level t), and these nodes will have higher perturbed log-probabilities G_{ϕ_S} than the other nodes at level t, which do not contain a top k leaf in the subtree. This means that if we discard all but the k nodes with highest log-probabilities G_{ϕ_S}, we are guaranteed to include the ancestors of the top k leaves. Formally, the k-th highest log-probability of the nodes at level t provides a *lower bound* required to be among the top k leaves, while G_{ϕ_S} is an *upper bound* for the set of leaves S such that it can be discarded or *pruned* if it is lower than the lower bound, so if G_{ϕ_S} is not among the top k.

Thus, when we apply the top-down sampling procedure, at each level we only need to expand the k nodes with the highest perturbed log-probabilities G_{ϕ_S} to end up with the top k leaves. By the Gumbel-Top-k trick the result is a sample without replacement from the sequence model. The effective procedure is a *beam search* over the (stochastically) perturbed log-probabilities G_{ϕ_S} for partial sequences $\boldsymbol{y}^S$, hence the name *Stochastic Beam Search*. As we use G_{ϕ_S} to select the top k partial sequences, we can also think of G_{ϕ_S} as the *stochastic score* of the partial sequence $\boldsymbol{y}^S$. We formalize Stochastic Beam Search in Algorithm 1.

Algorithm 1 StochasticBeamSearch($p_{\boldsymbol{\theta}}, k$)

1: **Input:** one-step probability distribution $p_{\boldsymbol{\theta}}$, beam/sample size k
2: Initialize BEAM empty
3: add ($\boldsymbol{y}^N = \varnothing, \phi_N = 0, G_{\phi_N} = 0$) to BEAM
4: **for** $t = 1, \ldots,$ steps **do**
5:   Initialize EXPANSIONS empty
6:   **for** $(\boldsymbol{y}^S, \phi_S, G_{\phi_S}) \in$ BEAM **do**
7:     $Z \leftarrow -\infty$
8:     **for** $S' \in$ Children(S) **do**
9:       $\phi_{S'} \leftarrow \phi_S + \log p_{\boldsymbol{\theta}}(\boldsymbol{y}^{S'} | \boldsymbol{y}^S)$
10:       $G_{\phi_{S'}} \sim \text{Gumbel}(\phi_{S'})$
11:       $Z \leftarrow \max(Z, G_{\phi_{S'}})$
12:     **end for**
13:     **for** $S' \in$ Children(S) **do**
14:       $\tilde{G}_{\phi_{S'}} \leftarrow -\log(\exp(-G_{\phi_S}) - \exp(-Z) + \exp(-G_{\phi_{S'}}))$
15:       add ($\boldsymbol{y}^{S'}, \phi_{S'}, \tilde{G}_{\phi_{S'}}$) to EXPANSIONS
16:     **end for**
17:   **end for**
18:   BEAM $\leftarrow$ take top k of EXPANSIONS according to $\tilde{G}$
19: **end for**
20: Return BEAM

3.2. Relation to Beam Search

Stochastic Beam Search should not only be considered as a sampling procedure, but also as a principled way to randomize a beam search. As a naive alternative, one could run an ordinary beam search, replacing the top-k operation by sam-

pling. In this scenario, at each step t of the beam search we could sample without replacement from the partial sequence probabilities $p_{\boldsymbol{\theta}}(\boldsymbol{y}_{1:t})$ using the Gumbel-Top-k trick.

However, in this naive approach, for a low-probability partial sequence to be extended to completion, it does not only need to be initially chosen, but it will need to be re-chosen, independently, again with low probability, at each step of the beam search. The result is a much lower probability to sample this sequence than assigned by the model. Intuitively, we should somehow *commit* to a sampling 'decision' made at step t. However, a hard commitment to generate exactly one descendant for each of the k partial sequences at step t would prevent generating any two sequences that share an initial partial sequence.

Our Stochastic Beam Search algorithm makes a *soft* commitment to a partial sequence (node in the tree) by propagating the Gumbel perturbation of the log-probability consistently down the subtree. The partial sequence will then be extended as long as its total perturbed log-probability is among the top k, but will fall off the beam if, despite the consistent perturbation, another sequence is more promising.

3.3. Relation to rejection sampling

As an alternative to Stochastic Beam Search, we can draw samples *without* replacement by rejecting duplicates from samples drawn *with* replacement. However, if the domain is large and the entropy low (e.g. if there are only a few valid translations), then rejection sampling requires many samples and consequently many (expensive) model evaluations. Also, we have to estimate how many samples to draw (in parallel) or draw samples sequentially. Stochastic Beam Search executes in a single pass, and requires computation linear in the sample size k and the sequence length, which (except for the beam search overhead) is equal to the computational requirement for sampling with replacement.

4. Experiments

4.1. Diverse Beam Search

In this experiment we compare Stochastic Beam Search as a principled (stochastic) alternative to *Diverse Beam Search* (Vijayakumar et al., 2018) in the context of neural machine translation to obtain a diverse set of translations for a single source sentence $\boldsymbol{x}$. Following the setup by Vijayakumar et al. (2018) we report both diversity as measured by the fraction of unique n-grams in the k translations as well as mean and maximum BLEU score (Papineni et al., 2002) as an indication of the quality of the sample. The maximum BLEU score corresponds to 'oracle performance' reported by Vijayakumar et al. (2018), but we report the mean as well since a single good translation and $k-1$ completely random sentences scores high on both maximum BLEU score and

diversity, while being undesirable. A good method should increase diversity without sacrificing mean BLEU score.

We compare four different sentence generations methods: *Beam Search* (BS), *Sampling*, *Stochastic Beam Search* (SBS) (sampling without replacement) and *Diverse Beam Search* with G groups (DBS(G)) (Vijayakumar et al., 2018). For Sampling and Stochastic Beam Search, we control the diversity using the softmax temperature T in Equation (5). We use $T = 0.1, 0.2, ..., 0.8$, where a higher T results in higher diversity. Heuristically, we also vary T for computing the scores with (deterministic) Beam Search. The diversity of Diverse Beam Search is controlled by the *diversity strengths* parameter, which we vary between $0.1, 0.2, ..., 0.8$. We set the number of groups G equal to the sample size k, which Vijayakumar et al. (2018) reported as the best choice.

We modify the Beam Search in `fairseq` (Ott et al., 2019) to implement Stochastic Beam Search[3], and use the `fairseq` implementations for Beam Search, Sampling and Diverse Beam Search. For theoretical correctness of the Stochastic Beam Search, we disable length-normalization (Wu et al., 2016) and early stopping (and therefore also do not use these parameters for the other methods). We use the pretrained model from Gehring et al. (2017) and use the `wmt14.v2.en-fr.newstest2014` [4] test set consisting of 3003 sentences. We run the four methods with sample sizes $k = 5, 10, 20$ and plot the minimum, mean and maximum BLEU score among the k translations (averaged over the test set) against the average $d = \frac{1}{4}\sum_{n=1}^{4} d_n$ of $1, 2, 3$ and 4-gram diversity, where n-gram diversity is defined as:

$$d_n = \frac{\text{\# of unique } n\text{-grams in } k \text{ translations}}{\text{total \# of } n\text{-grams in } k \text{ translations}}.$$

In Figure 2, we represent the results as curves, indicating the trade-off between diversity and BLEU score. The points indicate datapoints and the dashed lines indicate the (averaged) minimum and maximum BLEU score. For the same diversity, Stochastic Beam Search achieves higher mean/maximum BLEU score. Looking at a certain BLEU score, we observe that Stochastic Beam Search achieves the same BLEU score as Diverse Beam Search with a significantly larger diversity. For low temperatures (< 0.5), the maximum BLEU score of Stochastic Beam Search is comparable to the deterministic Beam Search, so the increased diversity does not sacrifice the best element in the sample. Note that Sampling achieves higher mean BLEU score at the cost of diversity, which may be because good translations are sampled repeatedly. However, the maximum BLEU score of both Sampling and Diverse Beam Search is lower than with Beam Search and Stochastic Beam Search.

---

[3]Our code is available at https://github.com/wouterkool/stochastic-beam-search

[4]https://s3.amazonaws.com/fairseq-py/data/wmt14.v2.en-fr.newstest2014.tar.bz2

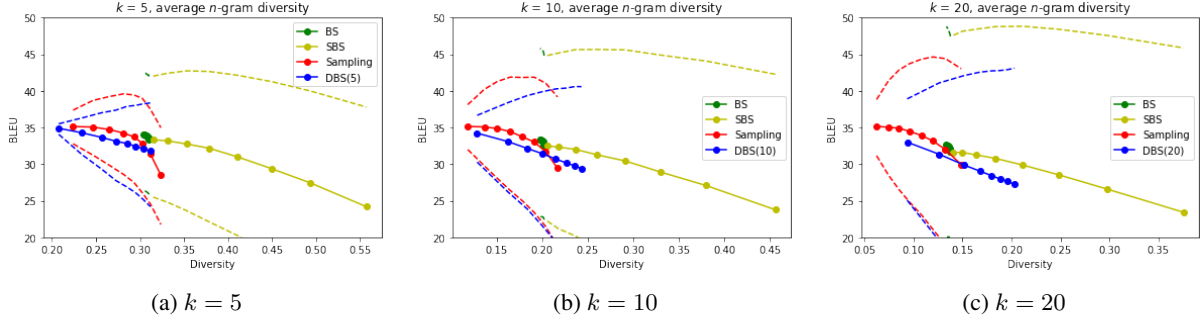(a) $k = 5$ (b) $k = 10$ (c) $k = 20$

Figure 2. Minimum, mean and maximum BLEU score vs. diversity for different sample sizes k. Points indicate different temperatures/diversity strengths, from 0.1 (low diversity, left in graph) to 0.8 (high diversity, right in graph).

4.2. BLEU score estimation

In our second experiment, we use sampling without replacement to evaluate the expected *sentence level* BLEU score for a translation $\boldsymbol{y}$ given a source sentence $\boldsymbol{x}$. Although we are often interested in *corpus level* BLEU score, estimation of sentence level BLEU score is useful, for example when training using minibatches to directly optimize BLEU score (Ranzato et al., 2016).

We leave dependence of the BLEU score on the source sentence $\boldsymbol{x}$ implicit, and write $f(\boldsymbol{y}) = \text{BLEU}(\boldsymbol{y}, \boldsymbol{x})$. Writing the domain of $\boldsymbol{y}$ (given $\boldsymbol{x}$) as $\boldsymbol{y}^1, ..., \boldsymbol{y}^n$ (e.g. all possible translations), we want to estimate the following expectation:

$$\mathbb{E}_{\boldsymbol{y} \sim p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})} [f(\boldsymbol{y})] = \sum_{i=1}^{n} p_{\boldsymbol{\theta}}(\boldsymbol{y}^i|\boldsymbol{x}) f(\boldsymbol{y}^i). \quad (11)$$

Under a Monte Carlo (MC) sampling *with replacement* scheme with size k, we write S as the set[5] of indices of sampled sequences $\{\boldsymbol{y}^i, i \in S\}$ and estimate (11) using

$$\mathbb{E}_{\boldsymbol{y} \sim p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})} [f(\boldsymbol{y})] \approx \frac{1}{k} \sum_{i \in S} f(\boldsymbol{y}^i). \quad (12)$$

If the distribution $p_{\boldsymbol{\theta}}$ has low entropy (for example if there are only few valid translations), then MC estimation may be inefficient since repeated samples are uninformative. We can use sampling without replacement as an improvement, but we need to use importance weights to correct for the changed sampling probabilities. Using the Gumbel-Top-k trick, we can implement an estimator equivalent to the estimator described by Vieira (2017), derived from priority sampling (Duffield et al., 2007):

$$\mathbb{E}_{\boldsymbol{y} \sim p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})} [f(\boldsymbol{y})] \approx \sum_{i \in S} \frac{p_{\boldsymbol{\theta}}(\boldsymbol{y}^i|\boldsymbol{x})}{q_{\boldsymbol{\theta},\kappa}(\boldsymbol{y}^i|\boldsymbol{x})} f(\boldsymbol{y}^i) \quad (13)$$

Here κ is the $(k+1)$-th largest element of $\{G_{\phi_i}, i \in N\}$, which can be considered the *empirical threshold* for the Gumbel-Top-k trick (since $i \in S$ if $G_{\phi_i} > \kappa$), and we define

---

[5]Formally, when sampling with replacement, S is a *multiset*.

$q_{\boldsymbol{\theta},a}(\boldsymbol{y}^i|\boldsymbol{x}) = P(G_{\phi_i} > a) = 1 - \exp(-\exp(\phi_i - a))$. If we would assume a fixed threshold a and *variably* sized sample $S = \{i \in N : G_{\phi_i} > a\}$, then $q_{\boldsymbol{\theta},a}(\boldsymbol{y}^i|\boldsymbol{x}) = P(i \in S)$ and $\frac{p_{\boldsymbol{\theta}}(\boldsymbol{y}^i|\boldsymbol{x})}{q_{\boldsymbol{\theta},a}(\boldsymbol{y}^i|\boldsymbol{x})}$ is a standard importance weight. Surprisingly, using a *fixed* sample size k (and empirical threshold κ) also yields in an unbiased estimator, and we include a proof adapted from Duffield et al. (2007) and Vieira (2017) in Appendix D. To obtain κ, we need to sacrifice the last sample[6], slightly increasing variance.

Empirically, the estimator (13) has high variance, and in practice it is preferred to normalize the importance weights by $W(S) = \sum_{i \in S} \frac{p_{\boldsymbol{\theta}}(\boldsymbol{y}^i|\boldsymbol{x})}{q_{\boldsymbol{\theta},\kappa}(\boldsymbol{y}^i|\boldsymbol{x})}$ (Hesterberg, 1988):

$$\mathbb{E}_{\boldsymbol{y} \sim p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})} [f(\boldsymbol{y})] \approx \frac{1}{W(S)} \sum_{i \in S} \frac{p_{\boldsymbol{\theta}}(\boldsymbol{y}^i|\boldsymbol{x})}{q_{\boldsymbol{\theta},\kappa}(\boldsymbol{y}^i|\boldsymbol{x})} f(\boldsymbol{y}^i). \quad (14)$$

The estimator (14) is biased but consistent: in the limit $k = n$ we sample the entire domain, so we have empirical threshold $\kappa = -\infty$ and $q_{\boldsymbol{\theta},\kappa}(\boldsymbol{y}^i|\boldsymbol{x}) = 1$ and $W(S) = 1$, such that (14) is equal to (11).

We have to take care computing the importance weights as depending on the entropy the terms in the quotient $\frac{p_{\boldsymbol{\theta}}(\boldsymbol{y}^i|\boldsymbol{x})}{q_{\boldsymbol{\theta},\kappa}(\boldsymbol{y}^i|\boldsymbol{x})}$ can become very small, and in our case the computation of $P(G_{\phi_i} > a) = 1 - \exp(-\exp(\phi_i - a))$ can suffer from catastrophic cancellation. For details, see Appendix C.

Because the model is not trained to use its own predictions as input, at test time errors can accumulate. As a result, when sampling with the default temperature $T = 1$, the expected BLEU score is very low (below 10). To improve quality of generated sentences we use lower temperatures and experiment with $T = 0.05, 0.1, 0.2, 0.5$. We then use different methods to estimate the BLEU score:

- *Monte Carlo* (MC), using Equation (12).

---

[6]For Stochastic Beam Search, we use a sample/beam size k, set κ equal to the k-th largest perturbed log-probability and compute (13) based on the remaining $k - 1$ samples. Alternatively, we could use a beam size of $k + 1$.
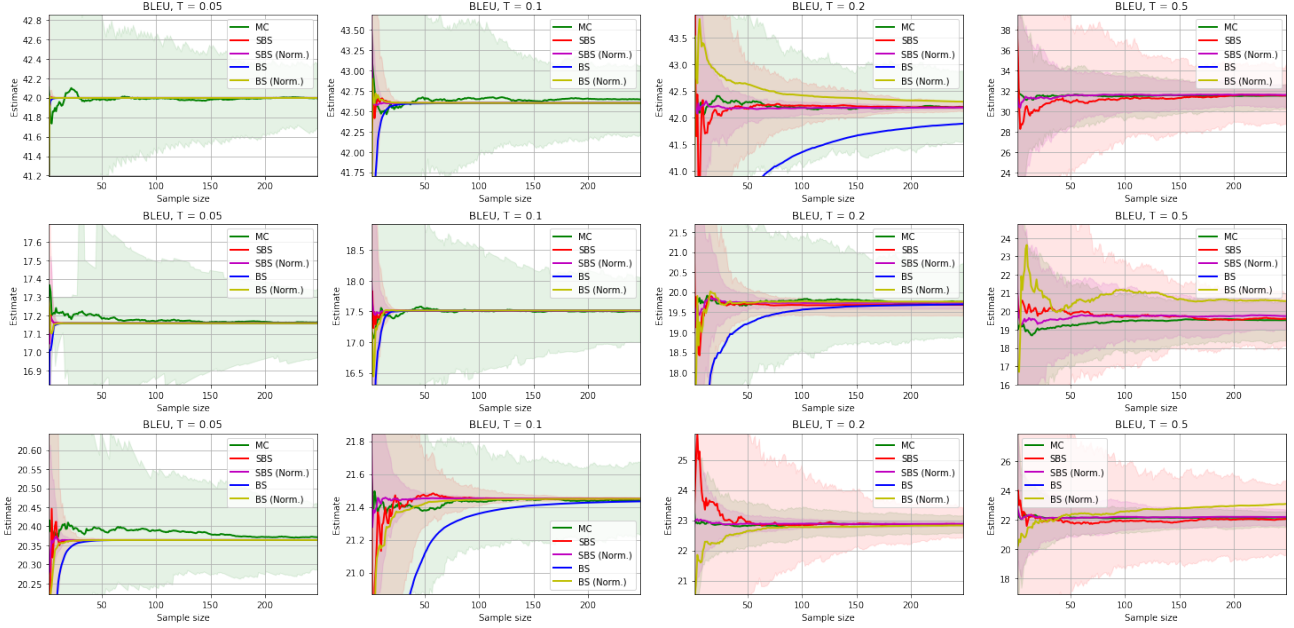
Figure 3. BLEU score estimates for three sentences sampled/decoded by different estimators for different temperatures.

- *Stochastic Beam Search* (SBS), where we compute estimates using the estimator in Equation (13) and the normalized variant in Equation (14).
- *Beam Search* (BS), where we compute a deterministic beam S (the temperature T affects the scoring) and compute $\sum_{i \in S} p_{\boldsymbol{\theta}}(\boldsymbol{y}^i|\boldsymbol{x}) f(\boldsymbol{y}^i)$. This is not a statistical estimator, but a lower bound to the target (11) which serves as a validation of the implementation and gives insight on how many sequences we need at least to capture most of the mass in (11). We also compute the normalized version $\frac{\sum_{i \in S} p_{\boldsymbol{\theta}}(\boldsymbol{y}^i|\boldsymbol{x}) f(\boldsymbol{y}^i)}{\sum_{i \in S} p_{\boldsymbol{\theta}}(\boldsymbol{y}^i|\boldsymbol{x})}$, which can heuristically be considered as a 'determinstic estimate'.

In Figure 3 we show the results of computing each estimate 100 times (BS only once as it is deterministic) for three different sentences[7] for temperatures $T = 0.05, 0.1, 0.2, 0.5$ and sample sizes $k = 1$ to 250. We report the empirical mean and 2.5-th and 97.5-th percentile. The normalized SBS estimator indeed achieves significantly lower variance than the unnormalized version and for $T < 0.5$, it significantly reduces variance compared to MC, without adding observable bias. For $T = 0.5$ the results are similar, but we are less interested in this case as the overall BLEU score is lower than for $T = 0.2$.

4.3. Conditional Entropy Estimation

Additionally to estimating the BLEU score we use $f(\boldsymbol{y}) = -\log p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})$ such that Equation (11) becomes the model

---

[7]Sentence 1500, 2000 and 2500 from the WMT14 dataset. Sentences 0, 500, 1000 are shorter and obtained 0 BLEU score in some cases, therefore being uninteresting in comparisons.

entropy (conditioned on the source sentence $\boldsymbol{x}$):

$$\mathbb{E}_{\boldsymbol{y} \sim p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})} \left[-\log p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) \right].$$

Entropy estimation is useful in optimization settings where we want to include an entropy loss to ensure diversity. It is a different problem than BLEU score estimation as high BLEU score (for a good model) correlates positively with model probability, while for entropy rare $\boldsymbol{y}$ contribute the largest terms $-\log p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})$. We use the same experimental setup as for the BLEU score and present the results in Figure 4. The results are similar to the BLEU score experiment: the normalized SBS estimate has significantly lower variance than MC for $T < 0.5$ while for $T = 0.5$, results are similar. This shows that Stochastic Beam Search can be used to construct practical statistical estimators.

5. Related Work

5.1. Sampling and the Gumbel-Max trick

The idea of sampling by solving optimization problems has been used for various purposes (Papandreou & Yuille, 2011; Hazan & Jaakkola, 2012; Tarlow et al., 2012; Ermon et al., 2013; Maddison et al., 2014; Chen & Ghahramani, 2016; Balog et al., 2017), but to our knowledge this idea has not been used for sampling without replacement.

Most closely related to our work, Maddison et al. (2014) note that the Gumbel-Max trick (Gumbel, 1954) can be applied implicitly and generalize it to continuous distributions, using an A^* search to find the maximum of a *Gumbel* process. In this work, we extend the idea of top-down sampling to efficiently draw *multiple samples without replacement*
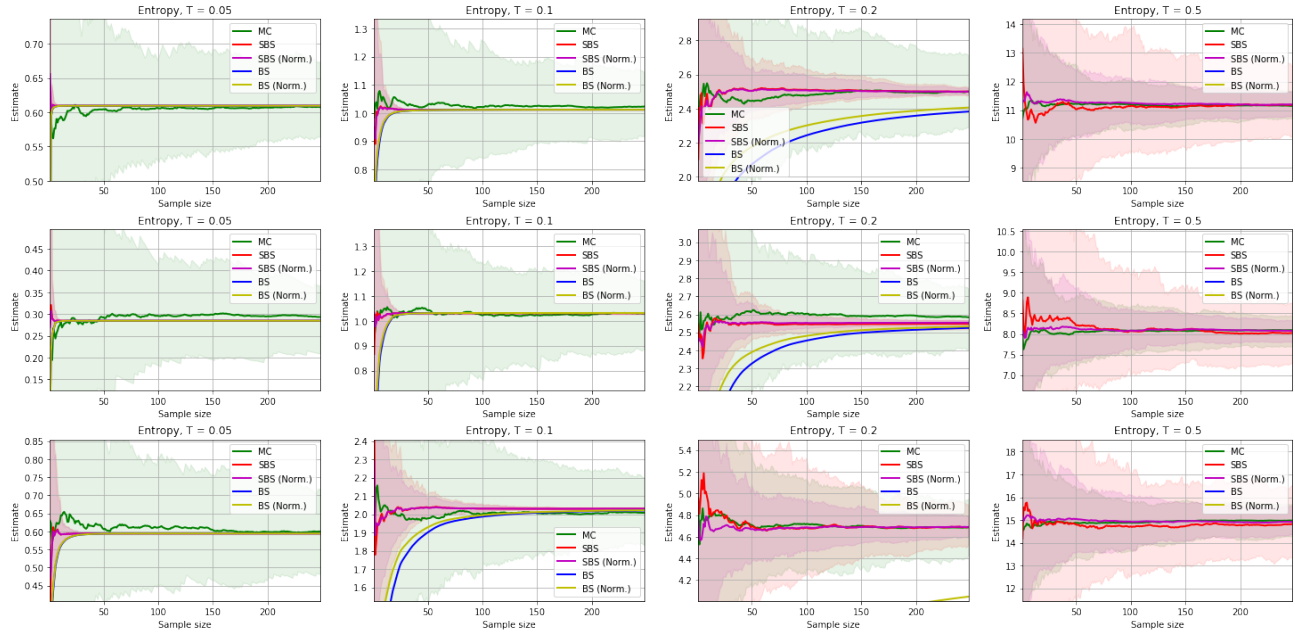
Figure 4. Entropy score estimates for three sentences sampled/decoded by different estimators for different temperatures.

from a *factorized distribution* (with possibly exponentially large domain) by implicitly applying the Gumbel-Top-k trick. This is a new and practical sampling method.

The blog post by Vieira (2014) describes the relation of the Gumbel-Top-k trick (as we call it) to Weighted Reservoir Sampling (Efraimidis & Spirakis, 2006), an algorithm for drawing weighted samples without replacement, either from a stream or efficiently in parallel. When sampling the complete domain, this is equivalent to the Thurstonian (Thurstone, 1927) interpretation of the Plackett-Luce ranking model (Plackett, 1975; Luce, 1959) as given by Yellott (1977). The connection of the Plackett-Luce model to the Gumbel-Max trick and the implication that this can be used for sampling without replacement is not widely known[8].

The Gumbel-Max trick has also been used to define relaxations of the categorical distribution (Maddison et al., 2016; Jang et al., 2016), which can be reparameterized for low-variance but biased gradient estimators. Recently, Grover et al. (2019) extended this idea to the Plackett-Luce distribution to optimize stochastic sorting networks. We think our work is a step in the direction to improve these methods in the context of sequence models (Gu et al., 2018).

5.2. Beam search

Beam search is widely used for approximate inference in various domains such as machine translation (Sutskever et al., 2014; Bahdanau et al., 2015; Ranzato et al., 2016; Vaswani et al., 2017; Gehring et al., 2017), image captioning

(Vinyals et al., 2015b), speech recognition (Graves et al., 2013) and other structured prediction settings (Vinyals et al., 2015a; Weiss et al., 2015). Although typically a test-time procedure, there are works that include beam search in the training loop (Daumé et al., 2009; Wiseman & Rush, 2016; Edunov et al., 2018b; Negrinho et al., 2018; Edunov et al., 2018a) for training sequence models on the sequence level (Ranzato et al., 2016; Bahdanau et al., 2017). Many variants of beam search have been developed, such as a continuous relaxation (Goyal et al., 2018), diversity encouraging variants (Li et al., 2016; Shao et al., 2017; Vijayakumar et al., 2018) or using modifications such as length-normalization (Wu et al., 2016) or simply applying noise to the output (Edunov et al., 2018a). Our Stochastic Beam Search is a principled alternative that shares some of the benefits of these heuristic variants, such as the ability to control diversity or produce randomized output.

6. Discussion

We introduced *Stochastic Beam Search*: an algorithm that is easy to implement on top of a beam search as a way to sample sequences without replacement. This algorithm relates sampling and beam search, combining advantages of these two methods. Our experiments support the idea that it can be used as a drop-in replacement in places where sampling or beam search is used. In fact, our experiments show Stochastic Beam Search can be used to yield lower-variance estimators and high-diversity samples from a neural machine translation model. In future work, we plan to leverage the probabilistic interpretation of beam search to develop new beam search related statistical learning methods.

---

[8]For example, currently the popular PyTorch (Paszke et al., 2017) library uses the (for large k) expensive sequential algorithm.

References

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, 2015.

Bahdanau, D., Brakel, P., Xu, K., Goyal, A., Lowe, R., Pineau, J., Courville, A., and Bengio, Y. An actor-critic algorithm for sequence prediction. In *International Conference on Learning Representations*, 2017.

Balog, M., Tripuraneni, N., Ghahramani, Z., and Weller, A. Lost relatives of the gumbel trick. In *International Conference on Machine Learning (ICML)*, pp. 371–379. PMLR, 2017.

Chen, Y. and Ghahramani, Z. Scalable discrete sampling as a multi-armed bandit problem. In *International Conference on Machine Learning*, pp. 2492–2501, 2016.

Daumé, H., Langford, J., and Marcu, D. Search-based structured prediction. *Machine learning*, 75(3):297–325, 2009.

Duffield, N., Lund, C., and Thorup, M. Priority sampling for estimation of arbitrary subset sums. *Journal of the ACM (JACM)*, 54(6):32, 2007.

Edunov, S., Ott, M., Auli, M., and Grangier, D. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 489–500, 2018a.

Edunov, S., Ott, M., Auli, M., Grangier, D., et al. Classical structured prediction losses for sequence to sequence learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pp. 355–364, 2018b.

Efraimidis, P. S. and Spirakis, P. G. Weighted random sampling with a reservoir. *Information Processing Letters*, 97(5):181–185, 2006.

Ermon, S., Gomes, C. P., Sabharwal, A., and Selman, B. Embed and project: Discrete sampling with universal hashing. In *Advances in Neural Information Processing Systems*, pp. 2085–2093, 2013.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*, pp. 1243–1252, 2017.

Goyal, K., Neubig, G., Dyer, C., and Berg-Kirkpatrick, T. A continuous relaxation of beam search for end-to-end training of neural sequence models. In *Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

Graves, A., Mohamed, A.-r., and Hinton, G. Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing, IEEE international conference on*, pp. 6645–6649, 2013.

Grover, A., Wang, E., Zweig, A., and Ermon, S. Stochastic optimization of sorting networks via continuous relaxations. In *International Conference on Learning Representations*, 2019.

Gu, J., Im, D. J., and Li, V. O. Neural machine translation with Gumbel-greedy decoding. In *Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

Gumbel, E. J. *Statistical theory of extreme values and some practical applications: a series of lectures*. Number 33. US Govt. Print. Office, 1954.

Hazan, T. and Jaakkola, T. On the partition function and random maximum a-posteriori perturbations. In *Proceedings of the 29th International Conference on Machine Learning*, pp. 1667–1674. Omnipress, 2012.

Hesterberg, T. C. *Advances in importance sampling*. PhD thesis, Stanford University, 1988.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2016.

Li, J., Monroe, W., and Jurafsky, D. A simple, fast diverse decoding algorithm for neural generation. *arXiv preprint arXiv:1611.08562*, 2016.

Luce, R. D. Individual choice behavior. 1959.

Mächler, M. Accurately computing $\log(1 - \exp(-|a|))$ assessed by the Rmpfr package, 2012. URL https://cran.r-project.org/web/packages/Rmpfr/vignettes/log1mexp-note.pdf.

Maddison, C. J., Tarlow, D., and Minka, T. A* sampling. In *Advances in Neural Information Processing Systems*, pp. 3086–3094, 2014.

Maddison, C. J., Mnih, A., and Teh, Y. W. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2016.

Negrinho, R., Gormley, M., and Gordon, G. J. Learning beam search policies via imitation learning. In *Advances in Neural Information Processing Systems*, pp. 10673–10682, 2018.

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., and Auli, M. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

Papandreou, G. and Yuille, A. L. Perturb-and-map random fields: Using discrete optimization to learn and sample from energy models. In *Computer Vision (ICCV), IEEE International Conference on*, pp. 193–200, 2011.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318, 2002.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in PyTorch. In *Advances in Neural Information Processing Systems*, 2017.

Plackett, R. L. The analysis of permutations. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 24(2):193–202, 1975.

Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations*, 2016.

Shao, Y., Gouws, S., Britz, D., Goldie, A., Strope, B., and Kurzweil, R. Generating high-quality and informative conversation responses with sequence-to-sequence models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2210–2219, 2017.

Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

Tarlow, D., Adams, R., and Zemel, R. Randomized optimum models for structured prediction. In *Artificial Intelligence and Statistics*, pp. 1221–1229, 2012.

Thurstone, L. L. A law of comparative judgement. *Psychological Review*, 34:278–286, 1927.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

Vieira, T. Gumbel-max trick and weighted reservoir sampling, 2014. URL https://timvieira.github.io/blog/post/2014/08/01/gumbel-max-trick-and-weighted-reservoir-sampling/.

Vieira, T. Estimating means in a finite universe, 2017. URL https://timvieira.github.io/blog/post/2017/07/03/estimating-means-in-a-finite-universe/.

Vijayakumar, A. K., Cogswell, M., Selvaraju, R. R., Sun, Q., Lee, S., Crandall, D. J., and Batra, D. Diverse beam search for improved description of complex scenes. In *AAAI*, 2018.

Vinyals, O., Fortunato, M., and Jaitly, N. Pointer networks. In *Advances in Neural Information Processing Systems*, pp. 2692–2700, 2015a.

Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer vision and Pattern Recognition*, pp. 3156–3164, 2015b.

Weiss, D., Alberti, C., Collins, M., and Petrov, S. Structured training for neural network transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pp. 323–333, 2015.

Wiseman, S. and Rush, A. M. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1296–1306, 2016.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Yellott, J. I. The relationship between luce's choice axiom, thurstone's theory of comparative judgment, and the double exponential distribution. *Journal of Mathematical Psychology*, 15(2):109–144, 1977.

A. Proof of the Gumbel-Top-k trick

Theorem 1. *For $k \leq n$, let $I_1^*, ..., I_k^* = \arg\operatorname{top} k \, G_{\phi_i}$. Then $I_1^*, ..., I_k^*$ is an (ordered) sample without replacement from the Categorical $\left(\frac{\exp \phi_i}{\sum_{j \in N} \exp \phi_j}, i \in N \right)$ distribution, e.g. for a realization $i_1^*, ..., i_k^*$ it holds that*

$$P\left(I_1^* = i_1^*, ..., I_k^* = i_k^*\right) = \prod_{j=1}^{k} \frac{\exp \phi_{i_j^*}}{\sum_{\ell \in N_j^*} \exp \phi_\ell} \qquad (15)$$

where $N_j^* = N \setminus \{i_1^*, ..., i_{j-1}^*\}$ is the domain (without replacement) for the j-th sampled element.

Proof. First note that

$$P\left(I_k^* = i_k^* \big| I_1^* = i_1^*, ..., I_{k-1}^* = i_{k-1}^*\right)$$

$$= P\left(i_k^* = \arg\max_{i \in N_k^*} G_{\phi_i} \big| I_1^* = i_1^*, ..., I_{k-1}^* = i_{k-1}^*\right)$$

$$= P\left(i_k^* = \arg\max_{i \in N_k^*} G_{\phi_i} \big| \max_{i \in N_k^*} G_{\phi_i} < G_{\phi_{i_{k-1}^*}}\right) \qquad (16)$$

$$= P\left(i_k^* = \arg\max_{i \in N_k^*} G_{\phi_i}\right) \qquad (17)$$

$$= \frac{\exp \phi_{i_k^*}}{\sum_{\ell \in N_k^*} \exp \phi_\ell}. \qquad (18)$$

The step from (16) to (17) follows from the independence of the max and $\arg\max$ (Section 2.3) and the step from (17) to (18) uses the Gumbel-Max trick. The proof follows by induction on k. The case $k = 1$ is the Gumbel-Max trick, while if we assume the result (15) proven for $k - 1$, then

$$P\left(I_1^* = i_1^*, ..., I_k^* = i_k^*\right)$$

$$= P\left(I_k^* = i_k^* \big| I_1^* = i_1^*, ..., I_{k-1}^* = i_{k-1}^*\right)$$

$$\cdot P\left(I_1^* = i_1^*, ..., I_{k-1}^* = i_{k-1}^*\right)$$

$$= \frac{\exp \phi_{i_k^*}}{\sum_{\ell \in N_k^*} \exp \phi_\ell} \cdot \prod_{j=1}^{k-1} \frac{\exp \phi_{i_j^*}}{\sum_{\ell \in N_j^*} \exp \phi_\ell} \qquad (19)$$

$$= \prod_{j=1}^{k} \frac{\exp \phi_{i_j^*}}{\sum_{\ell \in N_j^*} \exp \phi_\ell}.$$

In (19) we have used Equation (18) and Equation (15) for $k - 1$ by induction. $\qquad \square$

B. Sampling set of Gumbels with maximum T

B.1. The truncated Gumbel distribution

A random variable G' has a *truncated* Gumbel distribution with location ϕ and maximum T (e.g. $G' \sim \text{TruncatedGumbel}(\phi, T)$) with CDF $F_{\phi,T}(g)$ if:

$$F_{\phi,T}(g)$$

$$= P(G' \leq g)$$

$$= P(G \leq g | G \leq T)$$

$$= \frac{P(G \leq g \cap G \leq T)}{P(G \leq T)}$$

$$= \frac{P(G \leq \min(g, T))}{P(G \leq T)}$$

$$= \frac{F_\phi(\min(g, T))}{F_\phi(T)}$$

$$= \frac{\exp(-\exp(\phi - \min(g, T)))}{\exp(-\exp(\phi - T))}$$

$$= \exp(\exp(\phi - T) - \exp(\phi - \min(g, T))). \qquad (20)$$

The inverse CDF is:

$$F_{\phi,T}^{-1}(u) = \phi - \log(\exp(\phi - T) - \log u). \qquad (21)$$

B.2. Sampling set of Gumbels with maximum T

In order to sample a set of Gumbel variables $\{\tilde{G}_{\phi_i} | \max_i \tilde{G}_{\phi_i} = T\}$, e.g. with their maximum being *exactly* T, we can first sample the $\arg\max$, i^* and then sample the Gumbels conditionally on both the max and $\arg\max$:

1. Sample $i^* \sim \text{Categorical}\left(\frac{\exp \phi_i}{\sum_j \exp \phi_j}\right)$. We do not need to condition on T since the $\arg\max$ i^* is independent of the max T (Section 2.3).

2. Set $\tilde{G}_{\phi_{i^*}} = T$, since this follows from conditioning on the max T and $\arg\max$ i^*.

3. Sample $\tilde{G}_{\phi_i} \sim \text{TruncatedGumbel}(\phi_i, T)$ for $i \neq i^*$. This works because, conditioning on the max T and $\arg\max$ i^*, it holds that:

$$P(\tilde{G}_{\phi_i} < g | \max_i \tilde{G}_{\phi_i} = T, \arg\max_i \tilde{G}_{\phi_i} = i^*, i \neq i^*)$$

$$= P(\tilde{G}_{\phi_i} < g | \tilde{G}_{\phi_i} < T).$$

Equivalently, we can let $G_{\phi_i} \sim \text{Gumbel}(\phi_i)$, let $Z = \max_i G_{\phi_i}$ and define

$$\tilde{G}_{\phi_i} = F_{\phi_i,T}^{-1}(F_{\phi_i,Z}(G_{\phi_i}))$$

$$= \phi_i - \log(\exp(\phi_i - T)$$

$$\quad - \exp(\phi_i - Z) + \exp(\phi_i - G_{\phi_i}))$$

$$= -\log(\exp(-T) - \exp(-Z) + \exp(-G_{\phi_i})). \qquad (22)$$

Here we have used (20) and (21). Since the transformation (22) is monotonically increasing, it preserves the $\arg\max$ and it follows from the Gumbel-Max trick (3) that

$$\arg\max_i \tilde{G}_{\phi_i} = \arg\max_i G_{\phi_i} \sim \text{Categorical}\left(\frac{\exp\phi_i}{\sum_j \exp\phi_j}\right).$$

We can think of this as using the Gumbel-Max trick for step 1 (sampling the argmax) in the sampling process described above. Additionally, for $i = \arg\max_i G_{\phi_i}$:

$$\tilde{G}_{\phi_i} = F^{-1}_{\phi_i,T}(F_{\phi_i,Z}(G_{\phi_i})) = F^{-1}_{\phi_i,T}(F_{\phi_i,Z}(Z)) = T$$

so here we recover step 2 (setting $\tilde{G}_{\phi_{i^*}} = T$). For $i \neq \arg\max_i G_{\phi_i}$ it holds that:

$$P(\tilde{G}_{\phi_i} \leq g | i \neq \arg\max_i G_{\phi_i})$$
$$= \mathbb{E}_Z(P(\tilde{G}_{\phi_i} \leq g | Z, i \neq \arg\max_i G_{\phi_i}))$$
$$= \mathbb{E}_Z(P(\tilde{G}_{\phi_i} \leq g | Z, G_{\phi_i} < Z))$$
$$= \mathbb{E}_Z(P(F^{-1}_{\phi_i,T}(F_{\phi_i,Z}(G_{\phi_i})) \leq g | Z, G_{\phi_i} < Z))$$
$$= \mathbb{E}_Z(P(G_{\phi_i} \leq F^{-1}_{\phi_i,Z}(F_{\phi_i,T}(g)) | Z, G_{\phi_i} < Z))$$
$$= \mathbb{E}_Z(F_{\phi_i,Z}(F^{-1}_{\phi_i,Z}(F_{\phi_i,T}(g))))$$
$$= \mathbb{E}_Z(F_{\phi_i,T}(g)) = F_{\phi_i,T}(g).$$

This means that $\tilde{G}_{\phi_i} \sim \text{TruncatedGumbel}(\phi_i, T)$, so this is equivalent to step 3 (sampling $\tilde{G}_{\phi_i} \sim \text{TruncatedGumbel}(\phi_i, T)$ for $i \neq i^*$).

B.3. Numeric stability of truncated Gumbel computation

Direct computation of (22) can be unstable as large terms need to be exponentiated. Instead, we compute:

$$v_i = T - G_{\phi_i} + \text{log1mexp}(G_{\phi_i} - Z) \qquad (23)$$
$$\tilde{G}_{\phi_i} = T - \max(0, v_i) - \text{log1pexp}(-|v_i|) \qquad (24)$$

where we have defined

$$\text{log1mexp}(a) = \log(1 - \exp(a)), \quad a \leq 0$$
$$\text{log1pexp}(a) = \log(1 + \exp(a)).$$

This is equivalent as

$$T - \max(0, v_i) - \log(1 + \exp(-|v_i|))$$
$$= T - \log(1 + \exp(v_i))$$
$$= T - \log\left(1 + \exp\left(T - G_{\phi_i} + \log\left(1 - \exp\left(G_{\phi_i} - Z\right)\right)\right)\right)$$
$$= T - \log\left(1 + \exp\left(T - G_{\phi_i}\right)\left(1 - \exp\left(G_{\phi_i} - Z\right)\right)\right)$$
$$= T - \log\left(1 + \exp\left(T - G_{\phi_i}\right) - \exp\left(T - Z\right)\right)$$
$$= -\log\left(\exp(-T) + \exp(-G_{\phi_i}) - \exp(-Z)\right)$$
$$= \tilde{G}_{\phi_i}$$

The first step can be easily verified by considering the cases $v_i < 0$ and $v_i \geq 0$. log1mexp and log1pexp can be computed accurately using $\text{log1p}(a) = \log(1 + a)$ and $\text{expm1}(a) = \exp(a) - 1$ (Mächler, 2012):

$$\text{log1mexp}(a) = \begin{cases} \log(-\text{expm1}(a)) & a > -0.693 \\ \text{log1p}(-\exp(a)) & \text{otherwise} \end{cases}$$

$$\text{log1pexp}(a) = \begin{cases} \text{log1p}(\exp(a)) & a < 18 \\ x + \exp(a) & \text{otherwise} \end{cases}$$

C. Numerical stability of importance weights

We have to take care computing the importance weights as depending on the entropy the terms in the quotient $\frac{p_{\boldsymbol{\theta}}(\boldsymbol{y}_i|\boldsymbol{x})}{q_{\boldsymbol{\theta}}(\boldsymbol{y}_i|\boldsymbol{x})}$ can become very small, and in our case the computation of $P(G_{\phi_i} > \kappa) = 1 - \exp(-\exp(\phi_i - \kappa))$ can suffer from catastrophic cancellation. We can rewrite this expression using the more numerically stable implementation $\text{exp1m}(x) = \exp(x) - 1$ as $p(G_{\phi_i} > \kappa) = -\text{exp1m}(-\exp(\phi_i - \kappa))$ but in some cases this still suffers from instability as $\exp(\phi_i - \kappa)$ can underflow if $\phi_i - \kappa$ is small. Instead, for $\phi_i - \kappa < -10$ we use the identity

$$\log(1 - \exp(-z)) = \log(z) - \frac{z}{2} + \frac{z^2}{24} - \frac{z^4}{2880} + \mathcal{O}(z^6)$$

to directly compute the log importance weight using $z = \exp(\phi_i - \kappa)$ and $\phi_i = \log p_{\boldsymbol{\theta}}(\boldsymbol{y}_i|\boldsymbol{x})$ (we assume ϕ_i is normalized):

$$\log\left(\frac{p_{\boldsymbol{\theta}}(\boldsymbol{y}_i|\boldsymbol{x})}{q_{\boldsymbol{\theta}}(\boldsymbol{y}_i|\boldsymbol{x})}\right) = \log p_{\boldsymbol{\theta}}(\boldsymbol{y}_i|\boldsymbol{x}) - \log q_{\boldsymbol{\theta}}(\boldsymbol{y}_i|\boldsymbol{x})$$
$$= \log p_{\boldsymbol{\theta}}(\boldsymbol{y}_i|\boldsymbol{x}) - \log\left(1 - \exp(-\exp(\phi_i - \kappa))\right)$$
$$= \log p_{\boldsymbol{\theta}}(\boldsymbol{y}_i|\boldsymbol{x}) - \log\left(1 - \exp(-z)\right)$$
$$= \log p_{\boldsymbol{\theta}}(\boldsymbol{y}_i|\boldsymbol{x}) - \left(\log(z) - \frac{z}{2} + \frac{z^2}{24} - \frac{z^4}{2880} + \mathcal{O}(z^6)\right)$$
$$= \log p_{\boldsymbol{\theta}}(\boldsymbol{y}_i|\boldsymbol{x}) - \left(\phi_i - \kappa - \frac{z}{2} + \frac{z^2}{24} - \frac{z^4}{2880} + \mathcal{O}(z^6)\right)$$
$$= \kappa + \frac{z}{2} - \frac{z^2}{24} + \frac{z^4}{2880} + \mathcal{O}(z^6)$$

If $\phi_i - \kappa < -10$ then $0 < z < 10^{-6}$ so this computation will not lose any significant digits.

D. Proof of unbiasedness of priority sampling estimator

The following proof is adapted from the proofs by Duffield et al. (2007) and Vieira (2017). For generality of the proof, we write $f(i) = f(\boldsymbol{y}^i)$, $p_i = p_{\boldsymbol{\theta}}(\boldsymbol{y}^i|\boldsymbol{x})$ and $q_i(\kappa) = q_{\boldsymbol{\theta}, \kappa}(\boldsymbol{y}^i|\boldsymbol{x})$, and we consider general keys h_i (not necessarily Gumbel perturbations).

We assume we have a probability distribution over a finite domain $1, ..., n$ with normalized probabilities p_i, e.g. $\sum_{i=1}^n p_i = 1$. For a given function $f(i)$ we want to estimate the expectation

$$\mathbb{E}[f(i)] = \sum_{i=1}^n p_i f(i).$$

Each element i has an associated random key h_i and we define $q_i(a) = P(h_i > a)$. This way, if we *know* the threshold a it holds that $q_i(a) = P(i \in S)$ is the probability that element i is in the sample S. As was noted by Vieira (2017), the actual distribution of the key does not influence the unbiasedness of the estimator but does determine the effective sampling scheme. Using the Gumbel perturbed log-probabilities as keys (e.g. $h_i = G_{\phi_i}$) is equivalent to the PPSWOR scheme described by Vieira (2017).

We define shorthand notation $h_{1:n} = \{h_1, ..., h_n\}$, $h_{-i} = \{h_1, ..., h_{i-1}, h_{i+1}, ..., h_n\} = h_{1:n} \setminus \{h_i\}$. For a given sample size k, let κ be the $(k+1)$-th largest element of $h_{1:n}$, so κ is the *empirical threshold*. Let κ'_i be the k-th largest element of h_{-i} (the k-th largest of all *other* elements).

Similar to Duffield et al. (2007) we will show that every element i in our sample contributes an unbiased estimate of $\mathbb{E}[f(i)]$, so that the total estimator is unbiased. Formally, we will prove that

$$\mathbb{E}_{h_{1:n}}\left[\frac{\mathbb{1}_{\{i \in S\}}}{q_i(\kappa)}\right] = 1 \tag{25}$$

from which the result follows:

$$\mathbb{E}_{h_{1:n}}\left[\sum_{i \in S} \frac{p_i}{q_i(\kappa)} f(i)\right]$$

$$= \mathbb{E}_{h_{1:n}}\left[\sum_{i=1}^n \frac{p_i}{q_i(\kappa)} f(i) \mathbb{1}_{\{i \in S\}}\right]$$

$$= \sum_{i=1}^n p_i f(i) \cdot \mathbb{E}_{h_{1:n}}\left[\frac{\mathbb{1}_{\{i \in S\}}}{q_i(\kappa)}\right]$$

$$= \sum_{i=1}^n p_i f(i) \cdot 1 = \sum_{i=1}^n p_i f(i) = \mathbb{E}[f(i)]$$

To prove (25), we make use of the observation (slightly rephrased) by Duffield et al. (2007) that conditioning on h_{-i}, we know κ'_i and the event $i \in S$ implies that $\kappa = \kappa'_i$ since i will only be in the sample if $h_i > \kappa'_i$ which means that κ'_i is the $k+1$-th largest value of $h_{-i} \cup \{h_i\} = h_{1:n}$. The reverse is also true (if $\kappa = \kappa'_i$ then h_i must be larger than κ'_i since otherwise the $k+1$-th largest value of $h_{1:n}$ will be smaller than κ'_i).

$$\mathbb{E}_{h_{1:n}}\left[\frac{\mathbb{1}_{\{i \in S\}}}{q_i(\kappa)}\right]$$

$$= \mathbb{E}_{h_{-i}}\left[\mathbb{E}_{h_i}\left[\frac{\mathbb{1}_{\{i \in S\}}}{q_i(\kappa)}\bigg| h_i\right]\right]$$

$$= \mathbb{E}_{h_{-i}}\left[\mathbb{E}_{h_i}\left[\frac{\mathbb{1}_{\{i \in S\}}}{q_i(\kappa)}\bigg| h_{-i}, i \in S\right] P(i \in S|h_{-i})\right.$$

$$\left. + \mathbb{E}_{h_i}\left[\frac{\mathbb{1}_{\{i \in S\}}}{q_i(\kappa)}\bigg| h_{-i}, i \notin S\right] P(i \notin S|h_{-i})\right]$$

$$= \mathbb{E}_{h_{-i}}\left[\mathbb{E}_{h_i}\left[\frac{1}{q_i(\kappa)}\bigg| h_{-i}, i \in S\right] P(i \in S|h_{-i}) + 0\right]$$

$$= \mathbb{E}_{h_{-i}}\left[\mathbb{E}_{h_i}\left[\frac{1}{q_i(\kappa)}\bigg| h_{-i}, i \in S\right] q_i(\kappa'_i)\right]$$

$$= \mathbb{E}_{h_{-i}}\left[\mathbb{E}_{h_i}\left[\frac{1}{q_i(\kappa)}\bigg| \kappa = \kappa'_i\right] q_i(\kappa'_i)\right]$$

$$= \mathbb{E}_{h_{-i}}\left[\mathbb{E}_{h_i}\left[\frac{1}{q_i(\kappa'_i)}\right] q_i(\kappa'_i)\right]$$

$$= \mathbb{E}_{h_{-i}}\left[\frac{1}{q_i(\kappa'_i)} q_i(\kappa'_i)\right] = \mathbb{E}_{h_{-i}}[1] = 1$$