TO SAMPFORD OR TO STANFORD: HOW TO FAIL IN ACADEMIA

Noè Canevascini, Julius Gruber, Lothar Heimbach, Kevin Zhang
ETH Zürich
D-MATH
{noec, jgruber, hlothar, kezhang}@ethz.ch

ABSTRACT

In machine translation we need a decoder which from an input starts generating text, Beam search is the most used algorithm used for this task. It returns set of k elements and their relative probabilities and gives us an intuition of the possible further evolutions of the sentence. However there are multiple issues associated with this algorithm. For example: as noted by [1] large beam widths result in sequences of tokens with probability distributions associated with low evaluation scores. Another possible issue is the deterministic nature of this algorithm, which makes it impossible to have diverse outputs or good estimators. [2] developed a stochastic beam search (SBS) algorithm based on the Gumbel-Top-k trick and [3] expanded on the idea replacing the sampling methods with Conditional Poisson Sampling (CPS). In this work we explore different sampling methods and algorithms (Sampford, Pareto and Bullshit-sampling) with the objective of bettering the ROGUE, BLEU and Sacre-BLEU scores. We failed miserably.

1. INTRODUCTION

Machine translation is an area of natural language processing (NLP) which relies on conditional language modelling. Conditional language modelling, when used in machine translations, aims to find the probability of an output sentence in a target language y given an input sentence in your starting language x.

$$P(y|x)$$

.

To choose a sequence of words, or tokens, nowadays we rely on Beam Search as developed by [4] or [5] and . The algorithm is understandable and easily applicable and has become the default decoder due to its good results in a broad range of applications as seen in [6] [7] [8] and [3].The closely related normal Greedy or Best-First search (technically Beam search with beamwidth equal to one) has often similar or worse performance as seen in [9]. In [9] we also see why beam search is such a good algorithm in our case, our search space is big and scales exponentially with the dictionary size. This is because we construct probability trees, with different possible paths. We look for the sentence incrementally, at each timestep ask ourselves what the next word could be, and then keep the k most probable paths until the END token is the most probable one. For visual reference see Figure 1 and Figure 2. In Figure 1 we notice that if we would calculate all possible different paths the two token layers would be fully connected, which would have prohibitive computational costs. Standard Beam-search can also be improved upon while still guaranteeing k-optimality in terms of decoding time and memory costs as done in [8] or introducing a some pruning techniques as done in [10].

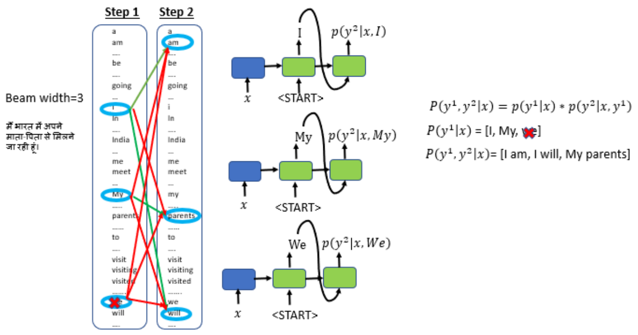

Fig. 1: Beam-Search in action with Beamwidth equal to three. Image from [11]

We saw that researchers ([3] [2] [13]) developed stochastic version of this deterministic algorithm, further details are in the section 2. The purpose of making Beam-Search stochastic is to have more diverse outputs. We give a statistical meaning to the algorithm through sampling. In this work we applied different sampling methods such as Sampford [14] and Pareto sampling to Beam-Search and compared the different strengths and weaknesses of the different sampling algorithms.

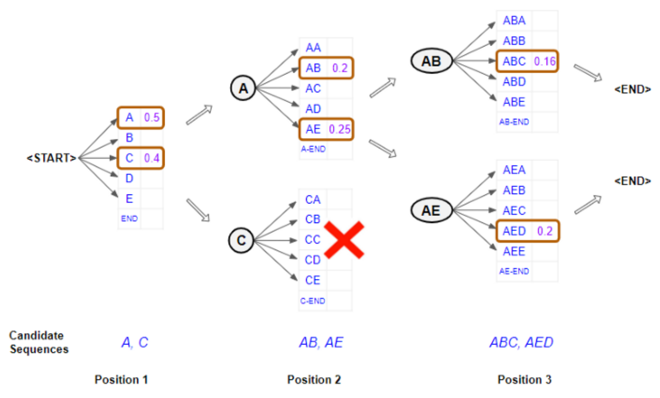

Fig. 2: Example of beam-search tree with beamwidth equal to two. Image from [12]

2. RELATED WORKS

In this section we discuss beam search and the different sampling methods currently used to try to make it a stochastic algorithm.

2.1. Beam Search

The idea of Beam-Search was published in [4] or [5]. The principle behind it is that instead of computing all possible conditional probabilities with all words (which would lead to prohibitively computational costs) we span a tree and only develop the most promising sub trees. Further details can be seen in 3.1. The ultimate objective is to find a sentence that maximizes the sequence model, which is a parametric distribution over sentences [2], [15].
This algorithm is clearly deterministic and will always lead to the same output if given the same input. So the task becomes to make beam search stochastic. In 2.2 we give an intuition of the possible implementations.

2.2. Stochastic versions of Beam Search

[2] propose SBS. This implementation is based on what they call the Gumbel-Top-k trick. The algorithm relies on the Gumbel-Max trick as explored in [16] and [17] with some slight modification such as sampling without replacement and taking a sample size of size k. The we explain the algorithm more formally in 3.2
[13] propose UniqueRandomizer. Their algorithm is closely related to SBS although mathematically less complex. UniqueRandomizer samples from randomized programs. The authors define randomized programs to be representations of factorizations of conditional distribuitons.
[3] propose Conditional Poisson SBS (CPSBS). This method, as the name suggests, is based on the conditional Poisson sampling scheme [18]. CPSBS samples without replacement replacing the top-K operator with the conditional poisson sampling. The details of the sampling method are explained in 3.3

3. MODEL

Here, we give a formal probabilistic setup behind Beam Search as well its stochastic pendant. As Beam Search has a wide range of possible applications, we first consider a more general framework. In section 4 we will then apply this model to specific problems. We base our setup on the previous work done in [2], [3] and [19].

3.1. Sequence Models

We work with a sequence model with bounded maximum sequence length $T > 0$. Formally, we consider a discrete probability space over the set of admissible outputs $\mathcal{Y} \subseteq V^T$ consisting of sequences over some vocabulary V. By filling up the words with a distinguished element of V, we assume that all outputs have the same length. We denote elements of $\mathcal{Y}$ as $\boldsymbol{y} = (y_n)_{1\leq nT}$, and write $\boldsymbol{y}_{<t}$ for the subsequence $(y_n)_{0\leq n\leq t}$. The space $\mathcal{Y}$ is equipped with some probability distribution p. Usually, this distribution is determined by the distributions of the y_t conditioned on $\boldsymbol{y}_{<t}$. The distribution of some random variable $\boldsymbol{Y} \sim p$ thus admits the representation

$$p(Y_1 = y_1, \dots, Y_t = y_t)$$

$$= p(Y_1 = y_1) \prod_{n=2}^{t} p(Y_n = y_n | Y_{<n} = y_{<n})$$

We then consider the maximization problem of p namely we are interested in some

$$y \in \arg\max_{y\in\mathcal{Y}} p(y)$$

As the size of $\mathcal{Y}$ is usually exponentially big, exact maximization is usually not computational feasible. Beam search is then used as a greedy-type approach that finds an approximate maximizer.
In this framework, Beam Search can be described as a sequence of sets $Y_1, ..., Y_T$ with Y_n consisting of sequences of length n. We denote the beam size by K. Additionally, for some $A \subseteq \mathcal{Y}$ and map $v : A \to \mathbb{R}$ we introduce the notation $k - \arg\max_{a\in A} v(a)$ as the set of all $B \subseteq A$ of size k, such that there is no $C \subseteq A$ with a bijective map $f : B \to C$ such that for all $b \in B$ we have $v(b) \leq v(f(b))$, and there is $b \in B$ such that $v(b) < v(f(b))$. Intuitively, $k - \arg\max_{a\in A} v(a)$ contains all consists of all possible subsets of A with k elements that have the arguments for the k highest values of v.
Then, Beam Search can be written inductively as

$$Y_1 = K - \arg\max_{v\in V} p(Y_1 = v)$$

and else

$$Y_n = K - \arg \max_{\boldsymbol{v} \in V^n | \boldsymbol{v}_{<n} \in V^{n-1}} p(Y_{\leq n} = v)$$

for $\boldsymbol{Y} \sim p$. As long as $\mathcal{Y}$ is big enough, the output of Beam Search is $Y_T \subseteq \mathcal{Y}$.
The idea of stochastic Beam Search is to randomize the Y_n. Instead of finding the $k - \arg\max$ in each step, we sample such subset according to a predetermined probability distribution. By this, the procedure becomes a stochastic process. This way, the output can also be used to find an estimator for e.g. the expected BLEU score.

3.2. Sampling

In this subsection we discuss how the sampling in each time step may be implemented. There are different approaches that differ in the resulting distribution of the output as well as in the properties of the sample as estimator.
Since in our context the goal is to find a probable subset of K elements, we are only interested in sampling without replacement with fixed sample size. Generally, a sampling method over some population $U = \{y_1, ..., y_N\}$ with N elements is then simply given by a probability distribution on the space of all subsets with K elements. To simplify the notation, we represent also subsets by their indicator functions, namely for $\boldsymbol{Y} \subseteq U$ we also write $Y_n = \chi_{y_n \in \boldsymbol{Y}}$. Let

$$S_k = \left\{ \boldsymbol{s} = (s_n)_{1 \leq n \leq N} \in \{0,1\}^N \mid \sum_{k=1}^{N} s_n = k \right\}$$

The sampling method is then a probability distribution Q on S_k.
There are different criteria on how to choose an appropriate distribution. For this, the inclusion probabilities π are important, which denote the probability of a given element being included in the sample set. Formally, we write $\pi(y_n) = Q(y_n \in Y)$, for a sample $Y \sim Q$. Note that π is no probability distribution over U, since

$$\sum_{n=1}^{N} \pi(y_n) = \mathbb{E}^Q \left[\sum_{n=1}^{N} \chi_{y_n \in Y} \right] = k$$

where χ denotes the indicator function. If we use the sample in an estimator, we attempt to choose the distribution to optimize our estimate.
One approach would be to simulate a typical draw by draw procedure, picking one element in each step without replacing it. After an element is drawn, the remaining probabilities will need to be conditioned by renormalizing them. This approach was analyzed in the original setting of Stochastic Beam Search in [2].

3.3. Conditional Poisson Sampling

Another approach is the Conditional Poisson Sampling design, which is a special exponential sampling design on S_k. The distribution is then dependent by some parameters $\boldsymbol{w} = (w_n)_{1 \leq n \leq N}$, given by

$$Q_{C,\boldsymbol{w},k}(\boldsymbol{Y}) = \alpha_{\boldsymbol{w}}^{-1} \prod_{n=1}^{N} w_n^{Y_n}$$

where $\alpha_{\boldsymbol{w},k} = \sum_{\boldsymbol{y} \in S_n} \prod_{n=1}^{N} w_n^{y_n}$ is a normalizing factor that cannot be further simplified into an explicit form. We will later discuss an appropriate choice of $\boldsymbol{w}$.
There is, however, a recursive method to sample efficiently, some typical methods can be found in [19].
The inclusion probabilities can also not be found in close form. To find them, one can use some fixed point iteration. Typically, there is a constellation of inclusion probabilities one is aiming for, the question is how to set the parameter w. However, this is very computational costly, especially if this has to be done in every step of Beam Search. Instead, in [8], they used an approximate implementation for Conditional Poisson Stochastic Beam Search. As it can be seen in [20], if one to achieve some fixed given inclusion probabilities $(\pi(y_n))_{1 \leq n \leq N}$, the choice $w_n = \frac{\pi(y_n)}{1-\pi(y_n)}$ gives a good approximation. In regard of the original maximization problem, the inclusion probabilities should then be proportional to $p(Y_{\leq n}) = v$ in the setup mentioned in 3.1.

3.4. Sampford Sampling

As the Sampling design will have an impact on the algorithm viewed as generalized Beam Search as well as a statistical estimator, we analyze new design to evaluate whether they result in significantly different performances. The first additional design we consider is Sampford Sampling as found in [19]. Here, the distribution is parametrized by inclusion probabilities $(\pi_n)_{1 \leq n \leq N}$ admitting $\sum_{n=1}^{N} \pi_n = k$. By writing $\omega_n = \frac{\pi_n}{1-\pi_n}$, the

$$Q_{S,\boldsymbol{\pi},k}(\boldsymbol{Y}) = \alpha_{\boldsymbol{\pi},k} \sum_{m=1}^{N} \pi_m Y_m \prod_{\substack{n=1 \\ n \neq m}}^{N} \omega_n^{Y_n}$$

where $\alpha_{\boldsymbol{\pi},k}$ is the corresponding normalizing factor. Although the representation of Sampford is more complex, and it is not an exponential method, Sampford admits an easy implementation as well. Furthermore, it bears the advantage that the inclusion probabilities can be exactly determined.
Building on the code of [3], we chose to implement Sampford sampling via the CPS rejective Sampford procedure, which is based on conditioning CPS of smaller sample size on a first sample according to a normalized version of the

inclusion probabilities. The algorithm is given by
In line 3, we relied on the implementation of Conditional

Algorithm 1: CPS rejective Sampford procedure

```
1: Sample some $y$ according to the categorical
   distribution given by the $\left(\frac{\pi_n}{k}\right)_{1\leq n\leq N}$
2: repeat
3:     Sample $\tilde{\boldsymbol{Y}} \sim Q_{C,\left(\frac{\pi_n}{1-\pi_n}\right)_{1\leq n\leq N},k-1}$ as in
       Conditional Poisson Sampling
4: until $y \notin \tilde{\boldsymbol{Y}}$
5: return $\tilde{\boldsymbol{Y}} \cup \{y\}$
```

Poisson Beam Search in [3], therefore. By this, we are implicitly using an approximate version as well.

3.5. Pareto sampling

3.6. Comparison

3.7. Statistical Estimation

4. EXPERIMENTAL RESULTS

The code was implement within the fairseq library [21] and run on both a CPU computing node from Euler and on a local machine with an AMD Ryzen 5950 X and a Nvidia GeForce RTX 3080 TI. We built our code on the existing code bases from [3], which itself builts upon [2]. We decided to implement our code base in a mixture of Cython vanilla Python.

5. FUTURE WORK

There are several ideas how this project can be extended. It would be interesting to implement the code in the current version of Fairseq so that it can be used by a wider audience of people. Furthermore, there would be the option to use more advance algorithms. The Monte-Carlo Tree Search algorithm, became famous in recent years in the deep learning community, with the use of it in the Alpha Go algorithm by Schrietwieser et. alli. This algorithm has not been applied to Beam Search in the NLP setting, still being applied here: ,https://dke.maastrichtuniversity.nl/m.winands/documents/CIG2012_paper_32.pdf. Another approach would be to look at the. We also thought about implementing a Quantum version of this algorithm. Unfortunately, we did not have enough qubits, because the freezer did not work. Also the pipeline caused a shortsqueeze of nonfungibletokens in the metaverse which were only solvable trough parabolic hyperbolic partially differentiable riemann problems with timestepping runge Kutta frogjump schemes when integrating by parts.

6. CONCLUSION

Yolo

7. REFERENCES

[1] Eldan Cohen and Christopher Beck, “Empirical analysis of beam search performance degradation in neural sequence models,” in *Proceedings of the 36th International Conference on Machine Learning*, Kamalika Chaudhuri and Ruslan Salakhutdinov, Eds. 09–15 Jun 2019, vol. 97 of *Proceedings of Machine Learning Research*, pp. 1290–1299, PMLR.

[2] Wouter Kool, Herke van Hoof, and Max Welling, “Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement,” 2019.

[3] Clara Meister, Afra Amini, Tim Viera, and Ryan Cotterell, “Conditional poisson stochastic beam search,” 2021.

[4] Alex Graves, “Sequence transduction with recurrent neural networks,” 2012.

[5] Carnegie-Mellon University.Computer Science Dept., “Speech understanding systems: summary of results of the five-year research effort at Carnegie-Mellon University.,” 4 2015.

[6] Iulian Vlad Serban, Tim Klinger, Gerald Tesauro, Kartik Talamadupula, Bowen Zhou, Yoshua Bengio, and Aaron Courville, “Multiresolution recurrent neural networks: An application to dialogue response generation,” 2016.

[7] Sergey Edunov, Myle Ott, Michael Auli, and David Grangier, “Understanding back-translation at scale,” in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium, Oct.-Nov. 2018, pp. 489–500, Association for Computational Linguistics.

[8] Clara Meister, Tim Vieira, and Ryan Cotterell, “Best-first beam search,” 2021.

[9] Christopher Wilt, Jordan Thayer, and Wheeler Ruml, “A comparison of greedy search algorithms,” 07 2010.

[10] Markus Freitag and Yaser Al-Onaizan, “Beam search strategies for neural machine translation,” *Proceedings of the First Workshop on Neural Machine Translation*, 2017.

[11] Renu Khandelwal, “An intuitive explanation of beam search,” 2020, [Online; accessed January, 2022].

[12] Ketan Doshi, “Foundations of nlp explained visually: Beam search, how it works,” 2021, [Online; accessed January, 2022].

[13] Kensen Shi, David Bieber, and Charles Sutton, “Incremental sampling without replacement for sequence models,” 2021.

[14] Y. Tille and Y. Tillé, *Sampling Algorithms*, Springer Series in Statistics. Springer, 2006.

[15] Wouter Kool, Herke van Hoof, and Max Welling, “Estimating gradients for discrete random variables by sampling without replacement,” in *International Conference on Learning Representations*, 2020.

[16] E.J. Gumbel, *Statistical Theory of Extreme Values and Some Practical Applications: A Series of Lectures*, Applied mathematics series. U.S. Government Printing Office, 1954.

[17] Chris J Maddison, Daniel Tarlow, and Tom Minka, “A* sampling,” in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, Eds. 2014, vol. 27, Curran Associates, Inc.

[18] Jaroslav Hajek, “Asymptotic Theory of Rejective Sampling with Varying Probabilities from a Finite Population,” *The Annals of Mathematical Statistics*, vol. 35, no. 4, pp. 1491 – 1523, 1964.

[19]

[20] LENNART BONDESSON, IMBI TRAAT, and ANDERS LUNDQVIST, “Pareto sampling versus sampford and conditional poisson sampling,” *Scandinavian Journal of Statistics*, vol. 33, no. 4, pp. 699–720, 2006.

[21] Alexei Baevskil Angela Fan Sam Gross Nathan Ng David Grangier Michael Auli Myle Ott4, Sergey Edunov, “FAIRSEQ: A Fast, Extensible Toolkit for Sequence Modeling,” .