Should you marginalize over possible tokenizations?

Nadezhda Chirkova[1] **Germán Kruszewski**[1] **Jos Rozen**[1] **Marc Dymetman**[2]
[1]Naver Labs Europe [2]Independent Researcher
{nadia.chirkova, german.kruszewski, jos.rozen}@naverlabs.com
marc.dymetman@gmail.com

Abstract

Autoregressive language models (LMs) map token sequences to probabilities. The usual practice for computing the probability of any character string (e.g. English sentences) is to first transform it into a sequence of tokens that is scored by the model. However, there are exponentially many token sequences that represent any given string. To truly compute the probability of a string one should *marginalize* over all tokenizations, which is typically intractable. Here, we analyze whether the practice of ignoring the marginalization is justified. To this end, we devise an importance-sampling-based algorithm that allows us to compute estimates of the marginal probabilities and compare them to the default procedure in a range of state-of-the-art models and datasets. Our results show that the gap in log-likelihood is no larger than 0.5% in most cases, but that it becomes more pronounced for data with long complex words.

1 Introduction

Language models are probability distributions over text strings. In practice, these distributions are defined over a vocabulary of *tokens*, such as words, punctuation marks, and other special symbols (Jurafsky, 2000; Goldberg, 2017). As long as a unique token sequence encodes any given string, the probability of a string according to the language model is equal to the probability of the corresponding token sequence. However, with today popular sub-word-level tokenizations this is not the case, as there are (exponentially) many possible tokenizations for any given string. For example, with the vocabulary $V = \{a, ab, b, c, ca, cab\}$, the string *"cab"* can be tokenized into $cab, c/a/b, ca/b, c/ab$. Therefore, the *true* probability that the language model assigns to the corresponding string is that obtained after marginalizing over *all possible tokenizations*. Yet, the common practice disregards this fact, computing the string probability by scoring a single *default* tokenization (e.g., cab). The implicit assumption from the community is that the probability mass of non-default tokenizations is negligible. However, this assumption has not been adequately evaluated yet.

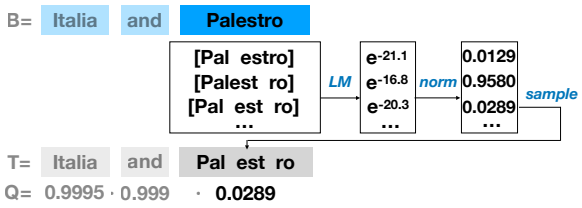

Figure 1: Illustration of the proposed procedure for sampling tokenization T and calculating its proposal probability $Q = Q(T|S)$ from a sequence of blocks B, produced by splitting sequence S.

In part, Cao and Rimell (2021) addressed this very same question, by conducting a pioneer study to quantify the gap between the default and marginalized probabilities. Their experiments with Transformer-XL pretrained on the WMT data (English and German) show negligible changes in perplexity with respect to using a single default tokenization for in-domain data and 0.9–1.9% improvement in perplexity for out-of-domain data, such as arXiv articles. Because exact marginalization is intractable in practice, marginalized probabilities were estimated using importance sampling. Importance sampling computes an unbiased estimate of the marginalized probabilities as an average over tokenizations sampled from a proposal distribution. Cao and Rimell (2021) exploited the probabilistic nature of the UnigramLM tokenizer (Kudo, 2018) to define such a proposal. As a consequence, their results do not necessarily extend to the more popular language models like GPT-2 (Radford et al., 2019), GPT-3 (Brown et al., 2020), BLOOM (Scao et al., 2022), T5 (Raffel et al., 2020), among others, trained using other tokenization schemes such as BPE (Sennrich et al., 2016), WordPiece (Schuster and Nakajima, 2012), among others.

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 2: Short Papers, pages 1–12

In this work, we devise a new proposal distribution that allows us to quantify the effect of marginalization for any given tokenizer. Equipped with this algorithm, we inspect the effect of marginalization over tokenizations for two LMs, GPT-2 (126M parameters, English) and the recently released BLOOM (1.7B parameters, multilingual), on various domains and languages. Our importance sampling estimates show that in practice marginalization does not influence log-likelihood much (usually less than 0.5% improvement), the highest influence (1–2% improvement) being for data with long, complex words and distribution shift. Because the results will vary for different models and data, we provide a tool for researchers and practitioners to measure the gap in their specific setting to decide whether the usual practice is warranted. To this end, we release our code[1], which can be applied to models from the `transformers` library.

2 Methodology

2.1 Preliminaries

Let us consider a sequence of characters S that we wish to score with an autoregressive language model P. Typically, S is split into a sequence $T = t_1, \ldots, t_n$ of tokens $t_i \in V$, where V is the model's vocabulary, a process commonly known as *tokenizing* the sequence. Then we can compute a score for a tokenization T of the sequence S, $P(T, S)$, using the chain rule:

$$P(T, S) = \mathbb{1}[T \to S] \prod_{j=1}^{|T|} P(t_j | t_{j-1}, \ldots, t_1)$$

where $T \to S$ indicates that T is a valid tokenization of S. Commonly used tokenization algorithms such as BPE or WordPiece provide a deterministic procedure for obtaining a *particular* way of tokenizing S into T, which we refer to as the *default* tokenization. Yet, in general, for the same sequence, there exist (exponentially) many possible tokenizations with vocabulary V, which also typically receive some probability mass by the LM. To obtain the *true* probability score for the sequence S, we should marginalize over all valid tokenizations: $P(S) = \sum_{T:T\to S} P(T, S)$.

However, computing $P(S)$ is typically intractable given the exponential number of valid tokenizations. Nonetheless, this value can be estimated through importance sampling, as follows. Introducing a proposal distribution $Q(T|S)$ over all tokenizations T of a sequence S, such that $P(T, S) > 0 \Rightarrow Q(T|S) > 0$, we can rewrite the probability $P(S)$, as follows:

$$P(S) = \sum_{T:T\to S} P(T, S) = \mathbb{E}_{Q(T|S)} \frac{P(T, S)}{Q(T|S)} \quad (1)$$

Now we can estimate $P(S)$ by sampling K independent tokenizations from the proposal:

$$P(S) \approx \frac{1}{K} \sum_{k=1}^{K} \frac{P(T_k, S)}{Q(T_k|S)}, \quad T_k \sim Q(T|S) \quad (2)$$

The quality of this estimate depends on the chosen proposal distribution: the closer the proposal $Q(T|S)$ is to the true posterior distribution $P(T|S)$, the smaller the variance of the unbiased estimate (2) tends to be.[2]

Algorithm 1 Proposal algorithm

Input: sequence S; max. block size L; max. number of tokenizations per block M
Output: a tokenization T sampled with prob. $Q(T|S)$

$T \leftarrow [\,]$; $q \leftarrow 1$
$B \leftarrow \text{split\_in\_blocks}(S, L)$
for $i = 1, \ldots, |B|$ **do**
  $X \leftarrow \text{get\_all\_tokenizations}(B_i, M)$
  for $j = 1, \ldots, |X|$ **do**
    $\hat{s}_j \leftarrow \text{LM}(X_j|T)$
  for $j = 1, \ldots, |X|$ **do**
    $s_j = \hat{s}_j / \sum_j \hat{s}_j$
  $j_* \leftarrow \text{sample}(s_1, \ldots, s_{|X|})$
  $T \leftarrow \text{concat}(T, X_{j_*})$
  $q \leftarrow q \cdot s_{j_*}$
$Q(T|S) \leftarrow q$
return $T, Q(T|S)$

2.2 Proposed approach

We introduce a novel proposal $Q(T|S)$ based on the LM itself with the intention to make it naturally closer to the posterior. Importantly, this proposal can be used for *any* tokenizer enabling its application to well-known state-of-the-art systems. The procedure for sampling from this proposal is presented in Algorithm 1 and also illustrated in Figure 1. In summary, the algorithm samples a tokenization T by building it incrementally as the concatenation of token subsequences T_i. Each token subsequence is sampled from the language model while

[1] https://github.com/naver/marginalization

[2] If we had access to the true posterior distribution $P(T|S)$, we would have $\frac{P(T,S)}{P(T|S)} = P(S)$, and therefore (i) one sample would be enough to obtain the needed value $P(S)$, and (ii) the variance of the importance sampling estimate would be zero.

always ensuring that the resulting tokenization is valid for the target S. To achieve this, the algorithm breaks S into a sequence of character blocks B, and *only* samples tokenizations T_i *that are valid for the corresponding block* B_i. Notably, in the extreme case of splitting S into a single block $B_1 = S$, our proposal $Q(T|S)$ turns into the true posterior $P(T|S)$, allowing to compute the exact marginalization with a single sample, as noted in footnote 2. However, because sampling a valid tokenization of a block requires renormalizing over *all* such valid tokenizations, this extreme instantiation would defeat the purpose of the algorithm as it would be equivalent to computing the full marginalization. Instead, we consider block sizes over which we can practically compute the renormalization constant by, for example, using whitespace-separated words as blocks. Still, because this can sometimes lead to impractically-sized blocks with a number of tokenizations that can exceed what we can reasonably score with a LM, we limit the maximum block size to a parameter L and we only score the top M block tokenizations inversely sorted by their number of tokens[3]. The resulting algorithm requires $O(|B| \times M)$ evaluations of the LM per-sample, where $|B|$ is the number of blocks used to split the sequence S. In Appendix E, we validate that, for short sentences with a tractable amount of possible tokenizations, for which we can actually compute the true value of the marginalization, our algorithm provides quite precise estimates.

3 Experiments

Experimental setup. We experiment with two language models, GPT-2 (Radford et al. 2019, 126M parameters, English) and the recently released BLOOM (Scao et al. 2022, 1.7B parameters, 45 natural and 12 programming languages). We select the following datasets for evaluating the LMs, which cover different styles and languages: Wikipedia articles (En), Twitter posts (En), CNN news (En), Transcriptions of White House Speeches (En), Flores-200 (sentences from Wikipedia in various languages, including high-resource, low-resource, latin and non-latin scripts), Python and C++ code (one recently released repository for each language). We concatenate texts into sequences of length 800 tokens (as measured by the default tokenization) to provide longer context for the LM. We evaluate on 100 sequences per dataset (Flores-200, CNN news and Code datasets are shorter). We refer to Appendix A for more details on the data and how we check that the LMs were not trained on the evaluation data.

[3]Appendices C and D describe practical details of these hyperparameters.

Exp.	$\mathrm{BPC}_{\mathrm{df}}$	$\mathrm{BPC}_{\mathrm{is}}$	BPC gap	% rel. gap	% ND
GPT-2 (125M params)					
Wiki	1.1076	1.1026	.0050	0.45%	0.9%
Twit	1.9610	1.9303	.0307	1.56%	4.2%
News	0.9421	0.939	.0028	0.30%	0.4%
Tr.sp.	1.0234	1.0029	.0204	1.99%	1.5%
BLOOM (1.7B params)					
Twit	1.7889	1.7653	.0236	1.32%	3.3%
News	0.8499	0.8462	.0037	0.55%	0.4%
Tr.sp.	0.9022	0.9002	.0020	0.23%	0.4%
Chi†	1.2080	1.2024	.0056	0.46%	3.1%
Fra	0.8001	0.7993	.0008	0.10%	0.2%
Spa	0.8813	0.8800	.0013	0.14%	0.3%
Vie	0.7939	0.7932	.0008	0.10%	0.1%
Ind	0.9812	0.9778	.0034	0.34%	0.6%
Eus	1.2432	1.2269	.0163	1.31%	3.5%
Urd†	0.8785	0.8697	.0088	1.00%	1.8%
Python	0.5100	0.5071	.0029	0.56%	1.3%
C++	0.6053	0.5993	.0059	0.98%	2.2%

Table 1: Main results. All natural languages from Flores-200 devtest set, sorted decreasingly by the size of corpora* used in BLOOM's training. † denotes non-latin script languages.

* Corpora sizes available at https://huggingface.co/bigscience/bloom#training-data

We measure the cross entropy (in BPC[4]) between the data and the model according to the default tokenization ($\mathrm{BPC}_{\mathrm{df}}$) and between the data and the marginalized model according to the importance sampling estimate ($\mathrm{BPC}_{\mathrm{is}}$), as well as their difference $\mathrm{BPC}_{\mathrm{df}} - \mathrm{BPC}_{\mathrm{is}}$ referred to as the BPC *gap*, and also the normalized difference $(\mathrm{BPC}_{\mathrm{df}} - \mathrm{BPC}_{\mathrm{is}})/\mathrm{BPC}_{\mathrm{df}}$ (relative BPC gap). Furthermore, we compute a 90% confidence interval $[\mathrm{BPC}_{\mathrm{is}}^{L}, \mathrm{BPC}_{\mathrm{is}}^{R}]$ around $\mathrm{BPC}_{\mathrm{is}}$, using bootstrap resampling (Wasserman, 2004, Chapter 8) for $n = 1000$ trials[5]. Additionally, we report the proportion of blocks for which our algorithm samples non-default tokenizations (%ND).

As for hyperparameters, we use $M = 128$ and choose L to be the maximum token length in the default tokenization of the evaluation data. We provide empirical validation for both these hyper-

[4]Bits Per Character: $\mathrm{BPC} = -(\log_2 \mathrm{Prob}(S))/|S|$, where Prob is the probability assigned to S: $P(T, S)$ in $\mathrm{BPC}_{\mathrm{df}}$ or $P(S)$ in $\mathrm{BPC}_{\mathrm{is}}$.

[5]We use the scipy.stats.bootstrap implementation, with `method='BCa'`.

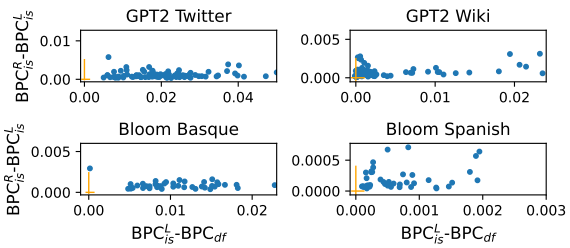

Figure 2: Confidence intervals visualisation. Each dot represents one data point (sequence).

parameters in Appendices D and C, respectively. We sample $K = 30$ tokenizations per sequence.

Results Table 1 presents our main results. We generally observe a low relative BPC gap ($< 0.5\%$), but in some cases exceeding 1%, e.g. 1.3–1.5% on Twitter, 2% on transcribed speech data, 1.3% on the Basque language (Eus) or 1% on the Urdu language (Urd). We note that dataset/model pairs with higher relative gap tend to be connected with low-resource languages (Basque and Urdu), non-latin scripts (Urdu and Chinese), and data distribution shift (transcribed speech, Twitter). Moreover, we observe a higher gap to be associated with a higher percentage of non-default tokenizations sampled by our algorithm (%ND). To learn more about the factors driving the probability of sampling the default tokenization, we bin blocks (which roughly correspond to words) from Wikipedia by the probability that our proposal assigns to their default tokenization, $Q(\mathrm{df.})$, when using GPT-2 as a model. Table 2 shows a few examples of blocks from each bin alongside the bin's frequency. As can be seen, high probability of sampling the default tokenization usually corresponds to common and simple words, whereas low probability corresponds to complex and rare words. From this observation, we conjecture that higher gaps are at least in part driven by the presence of long complex words in the datasets.

Finally, Figure 2 visualizes confidence intervals on BPC gaps for *individual* sequences across several datasets. Additional results are given in Appendix F. In particular, we plot the left limit of the confidence interval for the BPC gap ($\mathrm{BPC}^L_{\mathrm{is}} - \mathrm{BPC}_{\mathrm{df}}$) on the x-axis and the width of the interval ($\mathrm{BPC}^R_{\mathrm{is}} - \mathrm{BPC}^L_{\mathrm{is}}$) on the y-axis (non-negative by definition). If a dot is located to the right of 0, it means that we are highly confident that the BPC gap is positive on that individual sequence. The farther the dot is on the x-axis, the higher the corresponding BPC gap is. Likewise, the lower the value on the y-axis, the lower is the variance of our estimate of the marginalized probability and, consequently, of the BPC gap. As can be seen, we obtain low-variance predictions for most of the sequences, and for almost all of them we can observe a positive BPC gap. Moreover, we can note a distributional difference between dataset/model pairs with a low BPC gap (such as those on the right-hand side of Figure 2, with points concentrated close to the 0 value) and those with high BPC gap (such as those represented on the left-hand side of Figure 2, with points spread up to the right).

$Q(\mathrm{df.})$	Freq.	Example blocks
>0.999	90%	Many, are, the, larger, amphibians, superficially, resemble
0.99–0.999	6.1%	crocodiles, whenever, bases, Rifenburg, sailed, precursors
0.9–0.99	2.2%	warships, propelled, Tomasz, redemption, Metoposaurus
0.5–0.9	0.7%	paedomorphic, Peltobatrachus, ironclad, Urabi, Tonnante
0–0.5	0.7%	temnospondyls, brevirostrine, Pugong, saurus, semiaquatic

Table 2: Examples of blocks binned by proposal probability of the default tokenization, with percentage of such blocks in the dataset. GPT-2 on Wikipedia data.

4 Related Work

Stochastic tokenization or marginalisation over tokenizations were widely investigated in the context of model *training* (Grave et al., 2019; van Merriënboer et al., 2017; Buckman and Neubig, 2018; Provilkov et al., 2020; Kudo, 2018) or learning better tokenizers (He et al., 2020); in contrast, we evaluate the effect of marginalization at the *inference* stage, when the tokenizer and the LM were trained in the default, commonly-used way. The closest study to ours is Cao and Rimell (2021), which relies on the stochastic version of the UnigramLM tokenizer as their proposal Q, and thus their approach is inapplicable to LMs with other tokenizers. They also had to introduce a set of heuristics such as imposing consistent tokenization of repeated words or enforcing the default tokenization to be included among the sampled tokenizations, to make this proposal closer to the posterior and to decrease the variance of importance sampling.

5 Conclusion

In this work, we have studied the effect of marginalization over possible tokenizations in language modeling. For this, we introduced a novel proposal distribution over tokenizations, which is used in the importance sampling algorithm to obtain estimates of the marginalized probability, and that can be applied to any tokenizer and language model. Our results show that the overall effect of marginalization over tokenizations is often smaller than 0.5%, although it becomes more pronounced for data with long complex words or distribution shift. We release our code to allow practitioners to check the effect of marginalization for their models of interest.

Limitations

The main limitation of the proposed approach is that it would be relatively costly to apply at production time, compared to the conventional LM evaluation. First, it requires drawing a number of tokenization samples, as defined by importance sampling, in contrast to a single pass through the evaluated sequence in the conventional approach. Second, the conventional approach can be conducted with teacher forcing and efficiently parallelized, while the proposed approach relies on block-by-block sequential processing. Nonetheless, the proposed algorithm is designed for analysis purposes rather than to be used in production systems, for which it is feasible to run it in a reasonable time, allowing users to evaluate the effect of marginalization for any tokenizer and language.

Broader impact

As the work is dedicated to evaluating existing models on publicly available datasets, we are not aware of any potential negative impact.

Acknowledgements

We would like to thank Matthias Gallé for his valuable feedback.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Jacob Buckman and Graham Neubig. 2018. Neural lattice language models. *Transactions of the Association for Computational Linguistics*, 6:529–541.

Kris Cao and Laura Rimell. 2021. You should evaluate your language model on marginal likelihood over tokenisations. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2104–2114, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loïc Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. 2022. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*.

Yoav Goldberg. 2017. *Neural network methods for natural language processing*, volume 10. Morgan & Claypool Publishers.

Edouard Grave, Sainbayar Sukhbaatar, Piotr Bojanowski, and Armand Joulin. 2019. Training hybrid language models by marginalizing over segmentations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1477–1482, Florence, Italy. Association for Computational Linguistics.

Xuanli He, Gholamreza Haffari, and Mohammad Norouzi. 2020. Dynamic programming encoding for subword segmentation in neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3042–3051, Online. Association for Computational Linguistics.

Dan Jurafsky. 2000. *Speech & language processing*. Pearson Education India.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. *arXiv preprint arXiv:1804.10959*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models.

Saif Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. Semeval-2018 task 1: Affect in tweets. In *Proceedings of the 12th international workshop on semantic evaluation*, pages 1–17.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2020. BPE-dropout: Simple and effective subword regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1882–1892, Online. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Rush Alexander M. Tow, Jonathan, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5149–5152. IEEE.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.

Bart van Merriënboer, Amartya Sanyal, Hugo Larochelle, and Yoshua Bengio. 2017. Multiscale sequence modeling with a learned dictionary.

Larry Wasserman. 2004. *All of statistics: a concise course in statistical inference*, volume 26. Springer.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

A Data

We consider the following datasets:

- Wikitext (https://huggingface.co/datasets/wikitext, wikitext-2-raw-v1 test subset, Merity et al., 2016, CC BY-SA 4.0 license);
- Twitter posts (https://huggingface.co/datasets/tweet_eval, emoji test subset, Mohammad et al., 2018);
- CNN News (https://www.kaggle.com/datasets/hadasu92/cnn-articles-after-basic-cleaning, CC0 license);
- The White House Speeches (https://www.kaggle.com/datasets/mohamedkhaledelsafty/the-white-house-speeches-and-remarks-12102022, CC0 license);
- Flores-200 (https://github.com/facebookresearch/flores/tree/main/flores200, Costa-jussà et al., 2022, CC BY-SA 4.0 license);
- Python Code (all .py files from https://github.com/naver/disco, Creative Commons Attribution-NonCommercial-ShareAlike 4.0 license);
- C++ Code (all .h and .cc files from https://github.com/microsoft/Triste, MIT license).

Wikitext and White House Speeches datasets consist of paragraphs extracted from Wikipedia articles (wikipedia.org) or from transcribed speeches. Flores-200 is composed of sentences extracted from English Wikipedia and translated by professional translators into 200 languages. Python and C++ Code data consists of code files. Twitter / News datasets consist of separate tweets / news articles. We compose sequences to evaluate an LM on, by concatenating texts listed above into sequences of 800 tokens according to the default tokenization (concatenated texts are separated by \n\n). The sequence always begins with a new text. Code and News data contains texts longer than 800 tokens, these texts are considered as separate sequences and clipped to 800 tokens. Table 3 reports statistics of the data. Maximum 100 sequences per dataset are considered (Flores-200 dataset, News dataset and code data are shorter).

Dataset	Av. / max length	Total # of sequences
Wikitext	98 / 556	100
Twitter	20 / 159	100
News	833 / 2940	63
Tr. sp.	33 / 158	100
Flores (En)	27 / 69	37
Python	320 / 2623	6
C++	2296 / 16324	12

Table 3: Data statistics. Average and maximal length: length of BLOOM's tokenization for raw texts from the dataset (before concatenation). For Flores we only list English as an example, as other language data is a translation of English sentences.

We checked that the data we evaluate on was not used in model training as follows. GPT-2 was not trained on Wikipedia data, as reported in its paper (Radford et al., 2019). BLOOM was trained on Wikipedia data, so we do not evaluate it on Wikipedia and English Flores data. At the same time, data for other languages is based on translations, which makes it safe to use it for evaluation. Twitter is not listed in data sources for GPT-2 (https://github.com/openai/gpt-2/blob/master/domains.txt) and BLOOM (https://huggingface.co/spaces/bigscience/BigScienceCorpus). For evaluation on code, we use the code of the libraries created after the BLOOM's training. Likewise, for evaluation on the news and White House speech data, we selected only texts released after 11.03.2022 (after the beginning of the largest BLOOM model's training).

B Additional information on experiments

The BLOOM model is released under the Responsible AI License, and GPT-2 is released under the Modified MIT License. Our code is based on the transformers library (Wolf et al., 2020) which is released under the Apache License 2.0 license. All assets allow usage for research purposes. Evaluation of the GPT-2 model was conducted on a single Tesla V-100 GPU (24–48 GPU hours per dataset), and evaluation of the BLOOM model conducted on a single Tesla A100 GPU (72–120 GPU hours per dataset).

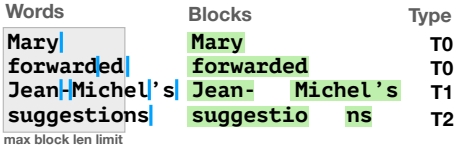

Figure 3: Illustration of blocks composition from whitespace- or newline-separated words. Vertical blue bars denote default tokenization. T1 and T2 blocks occur for words longer than the maximum block length L, here equal to 9.

	L	%T2	%T1	BPC gap	% $\text{BPC}_{\text{is}} < \text{BPC}_{\text{df}}$
French	17	0.08	0.5	-0.00108	77
	19	0	0.33	0.00084	100
	21	0	0.08	0.00079	98
Urdu	19	0.3	1.3	0.0019	62
	21	0	0.3	0.0087	100
	23	0	0.2	0.0089	100

Table 4: Effect of the maximum block length hyperparameter, L, on the portion of T1 and T2 blocks, BPC gap and the portion of sequences with $\text{BPC}_{\text{is}} < \text{BPC}_{\text{df}}$. Model: BLOOM-1.7B.

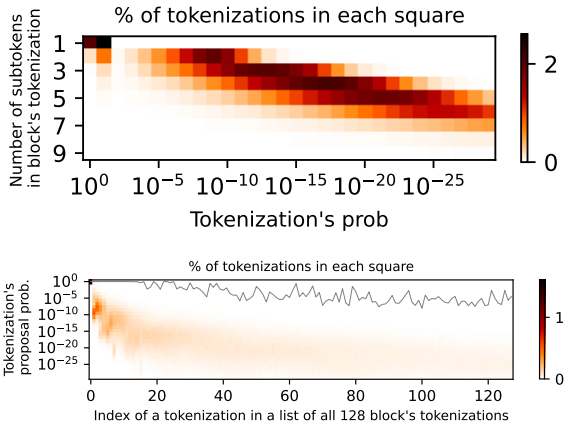

Figure 4: Top: the correlation between the number of subtokens in a block's tokenization and tokenization's proporal probability. Bottom: visualization of the proposal probabilities of blocks' tokenizations versus their ranks by the number of subtokens. Gray line specifies the maximum probability seen for each rank, which does not exceed 10^{-2} for ranks greater than 80. Both plots for GPT-2 on English Flores data.

C Segmentation into blocks

As discussed in Section 2.2, the proposal algorithm splits the sequence into a sequence of blocks. In our experiments, we split the sequence at white spaces and new line characters, thus making blocks roughly correspond to words. Because our algorithm computes all possible tokenizations within a block, this process can become prohibitively expensive for long blocks, which can occur with complex words or in languages that do not frequently use the white space character, such as Chinese. For this reason, we define a *maximum block length* hyperparameter, L. Words that have length lower or equal to L are denoted as type 0 (T0) blocks. If a word has length larger than L, it is split into smaller blocks, as follows. First, we compute the block's default tokenization and incrementally merge the tokens while checking not to exceed L. Once the limit is reached, a new block is started. The resulting blocks are denoted as type 1 (T1) blocks. Suppose at any point a token of length larger than L is encountered. In that case, this token is cropped at L, and the remaining characters are then moved to a new block. These blocks are denoted as type 2 (T2) blocks. Figure 3 illustrates these three block types.

Table 4 illustrates the effect that the maximum block length hyperparameter L has for BLOOM on French (low-gap case) and Urdu (higher-gap case). We experiment with three values of L to represent various proportions of T1 and T2 blocks. For low values of L ($L = 17$ and $L = 19$ for French and Urdu, respectively), we observe some small or even negative gap in BPC, and a large percentage of sequences that have higher cross-entropy when using the marginal than when using the default tokenization. This result comes with a small but non-negligible percentage of T2 blocks. Because T2 splits a token that is selected by the default tokenization across different blocks, this prevents the proposal from ever sampling the default tokenization, resulting in a poor estimate. Higher values of L result in the elimination of any T2 blocks with also a moderate impact on T1 blocks. Yet, once T2 blocks are eliminated, the number of T1 blocks does not appear to have a sizeable effect. Overall, these results provide the rule for selecting L: it should be set to the maximum length of the tokens in the default tokenization of the evaluation data in order to avoid T2 blocks.

D Limiting the number of tokenizations per block

The proposed importance sampling algorithm limits M, the number of tokenizations per block which are scored with LM, for better efficiency. In this section we motivate why it is not harmful for the results. In the top plot of Figure 4 we show that the proposal probability of a block's tokenization

$\mathrm{BPC}_{\mathrm{df}} - \mathrm{BPC}_{\mathrm{is}}$	$\mathrm{BPC}_{\mathrm{is}} - \mathrm{BPC}_{\mathrm{m}}$	$\mathrm{BPC}_{\mathrm{is}}^{L} - \mathrm{BPC}_{\mathrm{m}}$	$\mathrm{BPC}_{\mathrm{is}}^{R} - \mathrm{BPC}_{\mathrm{m}}$
N1: “runspiration from quotes”			
.036271	-.000479	-.001765	.000529
N2: “Did organgatuangs fly”			
.023346	.000675	-.000129	.001397
N3: “the Buffalo-Pitt road”			
.000530	.000313	-.000293	.000773
N4: “It’s Friday today”			
.000009	.000003	.000003	.000003
N5: “throughput of 600Mbit/s”			
.000170	-.000001	-.000001	-.000001
N6: “Television morning show”			
.000304	-.000024	-.000024	-.000024
N7: “snowboarding is cool”			
-.000136	.000219	.000038	.0004613

Table 5: Comparison of the proposed algorithm and exact marginalization for short sentences. We expect the value in the first column to be positive, the second value to be close to 0, the third value to be close to 0 and negative and the fourth value to be close to 0 and positive (the last two conditions mean the conf. interval on $\mathrm{BPC}_{\mathrm{is}}$ includes $\mathrm{BPC}_{\mathrm{m}}$).

strongly correlates with the number of subtokens in the tokenization. This motivates selecting top-M tokenizations per block by sorting the block's tokenizations by the decreasing number of subtokens (we use $M = 128$ in our experiments). Now, in the bottom plot of Figure 4 we present the 2d-histogram of proposal probabilities of blocks' tokenizations and their ranks in the sorting. It can be seen that the proposal probabilities of tokenizations with ranks higher than 10 have very low probabilities, i.e. usually lower than 10^{-10}. In fact, tokenizations with ranks greater than 70 were never sampled (in 99.95% one of the first 10 tokenizations was sampled, in 0.05% cases — one of tokenizations with indices 11–40, and in 0.0004% cases — with ranks 40–69.).

E Algorithm validation on short sentences

To validate the proposed algorithm, we compare the marginal BPC estimated with our algorithm to the true marginal BPC, $\mathrm{BPC}_{\mathrm{m}}$, obtained by enumerating all tokenizations of several relatively short sentences ($\leqslant 25$ characters, < 1M tokenizations). From Table 5 we observe that for sentences with relatively high BPC gap (N1–2), our estimate $\mathrm{BPC}_{\mathrm{is}}$ is close to $\mathrm{BPC}_{\mathrm{m}}$, with a thin confidence interval

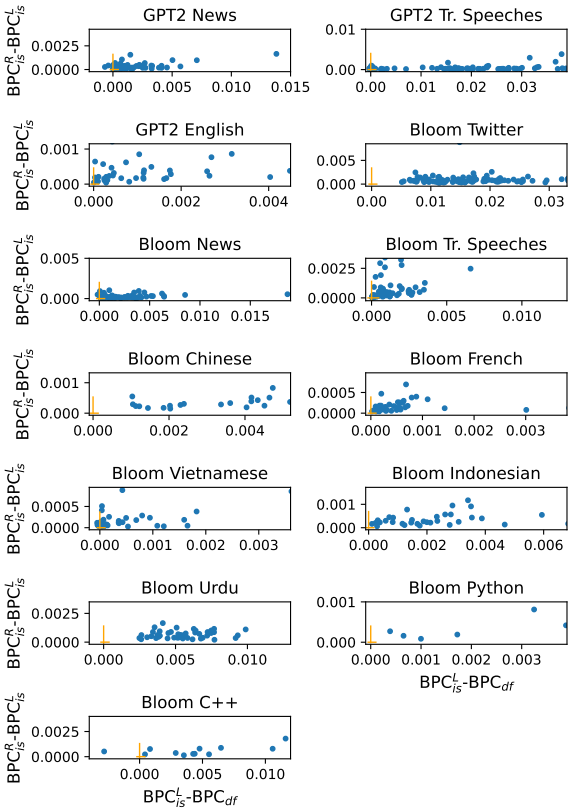

Figure 5: Confidence intervals visualisation. Each dot represents one data point (sequence). Please mind different axes scales.

which includes $\mathrm{BPC}_{\mathrm{m}}$. N3 shows the case with lower BPC gap, for which our estimate $\mathrm{BPC}_{\mathrm{is}}$ is between $\mathrm{BPC}_{\mathrm{df}}$ and $\mathrm{BPC}_{\mathrm{m}}$, and the confidence interval is wider but still includes $\mathrm{BPC}_{\mathrm{is}}$. N4–6 show the case of low BPC gap, in which our proposal always sampled the default tokenization hence there is no variance. In all three cases the difference between our estimate ($\mathrm{BPC}_{\mathrm{is}}$) and the marginal ($\mathrm{BPC}_{\mathrm{m}}$) is 3–100 times smaller than between default ($\mathrm{BPC}_{\mathrm{df}}$) and marginal ($\mathrm{BPC}_{\mathrm{m}}$). Finally, N7 shows the case with low BPC gap, in which our proposal did sample some non-default tokenizations, and the resulting estimate $\mathrm{BPC}_{\mathrm{is}}$ was larger than $\mathrm{BPC}_{\mathrm{df}}$. However, this ordering almost never happens with long texts, which is the intended use-case of our algorithm. To summarize, in almost all cases our algorithm produces a quite precise estimate.

F Additional confidence interval plots

Figure 5 shows additional confidence interval plots. The conclusions are the same as for plots in the main text.

Block's frequency	>=1e-4	<1e-4
% such blocks	0.602	0.398
% sampled default tokenizations	0.978	0.925
% sampled non-default tokenizations	0.022	0.075
% sampled length-1 tokenizations*	0.829	0.306
% sampled length-2 tokenizations*	0.137	0.299
% sampled length>=3 tokenizations*	0.034	0.395

Table 6: Additional analysis: the distribution of number of tokens in sampled block's tokenization, for low-frequency and high-frequency blocks, for GPT-2 on Twitter data. Rows denoted with * include both default and non-default tokenizations.

G Additional analysis

The intuition why the impact of non-default tokenizations becomes more pronounced for complex words, low-resource languages and data distribution shift is that all these cases are characterized by the appearance of blocks which were rarely or never seen during training. Roughly speaking, frequent words are encoded with short token sequences (1-2 tokens) by design of the tokenizer. Furthermore, the language model assigns high probability to the default tokenizations of these words because it saw them frequently during training. As a result, the effect of marginalization is small. In contrast, rare words are encoded with longer token sequences, and because they are not frequently seen during training, the language model can assign high probabilities to other tokenizations than a default one.

To illustrate given reasoning, Table 6 reports the distribution of number of tokens in sampled block's tokenization, for low-frequency and high-frequency blocks, for GPT-2 on Twitter data. Low-frequency blocks are split into more tokens and have a higher portion of non-default tokenizations.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Limitations

☑ A2. Did you discuss any potential risks of your work?
Broader impact

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract and introduction

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 3 and Appendices A, B

☑ B1. Did you cite the creators of artifacts you used?
Section 3 and Appendices A, B

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Appendices A, B

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Appendix B

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Our research is devoted to evaluating log-likelihood of existing models, we do not release any new models or textual artefacts. That is why we do not anticipate any harms from our work.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Section 3 and Appendix A

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Appendix A

C ☑ Did you run computational experiments?

Section 3 and Appendices

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 3 and Appendix B

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 3 and Appendices C, D

☐ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Not applicable. Left blank.

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Appendix B

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?
Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Not applicable. Left blank.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Not applicable. Left blank.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Not applicable. Left blank.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. Left blank.