International Journal of Internet, Broadcasting and Communication Vol.12 No.2 45-50 (2020)
http://dx.doi.org/10.7236/IJIBC.2020.12.2.45

IJIBC 20-2-7

Subword Neural Language Generation with Unlikelihood Training

Salahuddin Muhammad Iqbal[1], Dae-Ki Kang[2]*

[1]*Master Student, Department of Computer Engineering, Dongseo University, Korea*
[2]*Professor, Department of Computer Engineering, Dongseo University, Korea*
salahuddin.mi@gmail.com, dkkang@dongseo.ac.kr

Abstract

A Language model with neural networks commonly trained with likelihood loss. Such that the model can learn the sequence of human text. State-of-the-art results achieved in various language generation tasks, e.g., text summarization, dialogue response generation, and text generation, by utilizing the language model's next token output probabilities. Monotonous and boring outputs are a well-known problem of this model, yet only a few solutions proposed to address this problem. Several decoding techniques proposed to suppress repetitive tokens. Unlikelihood training approached this problem by penalizing candidate tokens probabilities if the tokens already seen in previous steps. While the method successfully showed a less repetitive generated token, the method has a large memory consumption because of the training need a big vocabulary size. We effectively reduced memory footprint by encoding words as sequences of subword units. Finally, we report competitive results with token level unlikelihood training in several automatic evaluations compared to the previous work.

Keywords: *Subword units, Natural language processing, Neural language generation, Natural Language processing Maximum likelihood training, Unlikelihood training.*

1. INTRODUCTION

Massive progress of deep learning models observed in many natural language generation tasks such as text summarization, dialogue response generation, and text generation. Conventionally, the model trained with maximum likelihood estimation objective, and the generated text produced approximated by decoding the sequence with top output probabilities of the model. However, the approach has been observed, giving monotonous content, and rarely using interesting words [1]. Furthermore, solely increasing the training data, purportedly does not address this problem [3]. Most of the implemented solutions are adjusted decoding strategies such as beam search [4, 6] or sampling strategies [1, 5]. These are only temporary solutions, while the main problem of the model's predicted output probabilities remains unsolved.

Welleck et al. [2] tried to address this issue by proposing a novel training objective by adding penalizing

Manuscript Received: February. 27, 2020 / Revised: March. 4, 2020 / Accepted: March. 10, 2020
Corresponding Author: dkkang@dongseo.ac.kr
Tel: +82-51-320-1724, Fax: +82-51-327-8955
Professor, Department of Computer Engineering, Dongseo University, Korea

candidate words in the existing likelihood objective. Incorrect repeating words and frequent words that often appear in the previous context restricted by this objective, which the process called unlikelihood training. Despite the candidate words are efficient to compute, implemented word tokenization was making the size of vocabulary bloated. The word tokenization, a process of splitting text document's words as smaller units called tokens, made the training required a massive amount of memory because of the large number of tokens' vector representation needed to be updated.

In this paper, we conducted experiments on decreasing the vocabulary size of the model by using subword tokenization. We trained the token level unlikelihood model with subword tokenization. Then, we compared the model with the existing approach, and it showed competitive results in several evaluation metrics. The paper organized as follows: Section 2 talks about methodologies, Section 3 explains the experiment setups, Section 4 shows the experiment results, and Section 5 concludes the paper.

2. METHODOLOGIES

2.1 Subword Tokenization

Subword tokenization is a tokenization method with subword segmentation of the sentence. Compared to word tokenization, which segmenting the sentence as a sequence of words split by spaces in between, subword tokenization sees words as sequent pair of characters. This tokenization is widely used in Neural Machine Translation systems [7, 8, 9] because of the advantages of keeping vocabulary size and number of unknown words minimum.

Byte-pair-encoding (BPE) [7] is a subword segmentation algorithm adapted from a compression algorithm [10]. BPE first initialize vocabulary with individual characters in the data. Then merge the most frequent pair of characters until reaching the desired vocabulary size |V|. The resulting vocabulary used to segment the input of the train, valid, test split. The tokenization represented rare words or symbols as a sequence of substring or characters. Consequently, only a small fixed size of the vocabulary was needed to encode text. Figure 1 shows the illustration of the segmentation of word tokenization and subword tokenization with a given input.

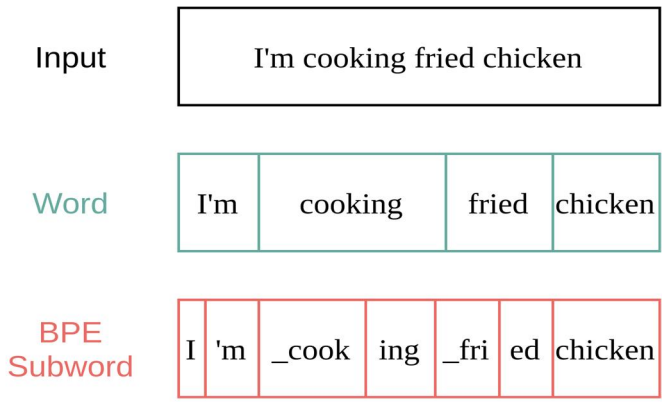

Figure 1. Illustration of word tokenization and BPE tokenization from the given input

2.2 Unlikelihood Training

In general, the standard approach to training a language model p_θ is maximum likelihood estimation (MLE), that try to minimize with a given sequence $X = \{x_1, x_2, \ldots, x_T\}$:

$$L_{MLE}(p_\theta, S) = -\sum_{i=1}^{|S|}\sum_{t=1}^{|x^{(i)}|} log\, p_\theta(x_t^{(i)} \mid x_{<t}^{(i)}) \quad (1)$$

where S is a set of samples, x is a sequence composed of tokens $x_t \in V$, and Vitself is defined vocabulary. A model $p_\theta$can be defined as a joint probability distribution over a sequence of tokens:

$$p_\theta(x) = \prod_{t=1}^{|x|} p_\theta(x_t \mid x_{<t}) \quad (2)$$

The main concept of unlikelihood training [2] is reducing the probability of the model of candidate tokens. Given a sequence $X = \{x_1, x_2, \ldots, x_T\}$ and a set of selected candidate tokens $C^t = \{c_1, \ldots, c_m\}$ for step t defined as in general form:

$$L_{UL}^t(p_\theta, C^t) = -\sum_{c \in C^t} log(1 - p_\theta(c \mid x_{<t})) \quad (3)$$

■ **Token Level Unlikelihood Training.** The training with token level unlikelihood involved the previous context tokens as candidate tokens excluding the true token. The candidate tokens in token level unlikelihood defined as:

$$C^t = \{x_1, x_2, \ldots, x_{t-1}\} \backslash \{x_t\} \quad (4)$$

According to Welleck et al. [2], minimizing Eq. (3) with this candidate set lowering the probability of frequent tokens and inaccurate repeating tokens. The overall objective for token level unlikelihood training then consists of MLE (1) and unlikelihood losses (3) in step t as:

$$L_{total}^t = L_{MLE}^t + \alpha\, L_{UL}^t \quad (5)$$

■ **Sequence Level Unlikelihood Training.** The candidate tokens in sequence level unlikelihood selected based on the repetitive tokens in generated sequences. The model generate a continuation $(x_{m+1}, x_{m+2}, \ldots, x_{m+M}) \sim p_\theta(\cdot \mid x_1, x_2, \ldots, x_m)$ given a prefix $(x_1, x_2, \ldots, x_m) \sim p_*$ and define the candidate tokens for $t \in \{m+1, m+2, \ldots, m+M\}$as:

$$C_{repeat-n}^t = \{x_t\} \text{ if } (x_{t-i}, x_{t-2}, \ldots, x_t, \ldots, x_{t+j}) \in x_{<t-i} \text{ for any } (j-i) = n, i \le n \le j \quad (6)$$

Finally, this training used objective Eq. (3) without likelihood objective Eq. (4). In this paper, we didn't perform the sequence level unlikelihood training due to memory constraints.

3. EXPERIMENTS

For our experiments, we use the subword tokenization with BPE to encode text and applied unlikelihood objectives to train huge neural language models, then used as text generators to complete sentences. We evaluate the models with evaluations in repetition and token-level mismatch and compared with the word-level model of unlikelihood.

3.1 Experiment Setups

We used Transformer architecture [11] with 16-layer, 8 attention heads, a vector embedding dimension of 1024, and a fully-connected dimension of 4096, based on Welleck et al. [2] implementation. We train each model for a maximum of 286,000 updates, evaluating on the validation set and selected model state with the best validation perplexity. We use the WikiText-103 dataset by Merity et al. [12], a huge collection of Wikipedia articles. We used BPE codes of 30k to train the BPE and apply tokenization to the dataset.

3.2 Evaluation Metrics

Evaluation metrics that we used are based on Welleck et al. [2] evaluations in repetition in token level (*rep*) and 4-grams sequence level (*seq-rep-4*), token distribution in token level (*uniq*) as well as sequence level (*uniq-seq*), and language modeling quality with perplexity (*ppl*). Also, we compared the number of parameters vector representations needed to be updated in each model.

- **Repetition.** Given a set *D* of length-*T* sequences, the evaluation for token-level repetition (*rep*) measures *top-1* predictions of next-token that occur in the previous *L* tokens is:

$$rep/L = \frac{1}{|D|T} \sum_{X \in D} \sum_{t=1}^{T} \mathbb{1}_{arg\ max\ p_\theta(x \mid X_{<t})} : X_{t-L-1:t-1} \tag{7}$$

Single-token repeat defined as a predicted token when $arg\ max\ p_\theta(x \mid X_{<t})$ is 1. For the sequence repetition evaluation, we use the subset of duplicate *n*-grams (*seq-rep-n*) in a generated sequence. Given a continuation $x_{m+1:m+M}$, the metric defined as:

$$\text{seq-rep-n} = 1.0 - \frac{|unique\ n-grams(x_{m+1:m+M})|}{|n-grams|} \tag{8}$$

and calculate the average over continuations. This metric increases towards 1.0 as the model repeats n-grams and zero when there is no repetition in the continuation. In this paper, we evaluated the model with 4-grams repetition metric (*seq-req-4*).

- **Token Distribution.** We measure the predicted token distribution by the model with the numbers of unique tokens. We reported the token level metric (*uniq*) with the number of unique next-token predictions on validation split *D*. As a sequence level metric (*uniq-seq*), we evaluate the number of unique tokens in continuations of validation split *D*.
- **Language Modeling Quality.** The accuracy (*acc*) of next-token prediction defined as $\frac{1}{N}|\{arg\ max\ p(x_t|x_{<t}) = x_t^* \mid x_{<t} \in D\}|$, with *N* prefixes $x_{<t}$ and true next tokens x_t^*. The perplexity (*ppl*) used to measure how well the model predicted the next token in validation split *D*. Generally, the perplexity defined as:

$$ppl = \sqrt[S]{\prod_{t=1}^{S} \frac{1}{p(x_t \mid x_{<t})}} \tag{9}$$

with S as the number of tokens in the validation split *D*.

4. RESULTS AND DISCUSSIONS

In the initial experiment, we compared the size of vocabulary from each tokenization. From Table 1, it can

be seen that the word tokenization has a significant amount of vocabulary compared to BPE tokenization. The result of BPE tokenization's vocabulary is 87% smaller compared to word tokenization. As a result, the model with BPE tokenization only needs to update around 36M parameters while the word tokenization requires to update 274M parameters. Also, we managed to train the model in the modern consumer GPU rather than industrial-grade GPUs used by Welleck et. al [2]. Despite vocabulary shrinking, the model recognized almost all the input in the valid and test split of the dataset without replacing it with the unknown token <*unk*>.

Table 1. Comparison of vocabulary size and tokenization unknown word replaced by <*unk*> token in train, valid, and test split of WikiText-103 dataset.

Tokenization	Vocabulary Size	Unknown Word		
		Train	Valid	Test
Word Tokenization	267,743	0.000%	0.000%	0.000%
BPE Tokenization	36,883	0.000%	0.020%	0.001%

In the second experiment, we evaluated the models with metrics as explained in subsection 3.2. Word tokenization models outperformed subword tokenization with BPE models' results. From Table 2, the word tokenization models trained with maximum likelihood ($Word_{MLE}$) and unlikelihood ($Word_{MLE}$)'s results are from Welleck et al. [2] model which is publicly available. $Subword_{MLE}$ and $Subword_{UL}$ are the subword tokenization model trained with maximum likelihood and unlikelihood objective respectively.

Table 2. Result for token-level objectives according to token-level metrics using the validation split of WikiText-103.

Model	rep	seq-rep-4	ppl	acc	uniq	uniq-seq
$Word_{MLE}$	0.619	0.429	24.592	0.401	11.6k	10.7k
$Word_{UL}$	0.569	0.274	25.624	0.396	12.4k	12.6k
$Subword_{MLE}$	0.598	0.487	21.929	0.439	13.3k	10.3k
$Subword_{UL}$	0.546	0.290	22.152	0.421	14.1k	12.2k

The result of the subword tokenization model with both training objectives is surprisingly compelling compared to the word tokenization models. Similar to $Word_{UL}$ repetition of token level *rep* and sequence level *seq-rep-4* of $Subword_{UL}$ result outperformed model that trained by maximum likelihood $Subword_{UL}$ with 0.052 and 0.197 difference of respective metric. The unlikelihood training $Subword_{UL}$ also improves unique token and sequence level generation performance (*uniq* 14.1k and *uniq-seq* 12.2k) compared to maximum likelihood training $Subword_{MLE}$ (*uniq* 13.3k and *uniq-seq* 10.3k). Despite the unlikelihood training $Subword_{UL}$ achieved improvement in several metrics compared to maximum likelihood training $Subword_{MLE}$, the model still relatively performed well in terms of perplexity (*ppl* 22.152 vs. 21.929).

Table 2 showed the effect of minimizing unlikelihood loss in repetition and token distribution metrics. The unlikelihood model with subword tokenization $Subword_{UL}$ successfully suppressed the repetitive tokens in generation. The results exhibited the effectiveness of the token level unlikelihood objective as discussed in Section 2.2.

Last but not least, we also included the result of $Word_{MLE}$ and $Word_{UL}$ in Table 2 only to emphasize that the reduction of vocabulary is not affecting the performance.

5. CONCLUSION

We successfully reduced vocabulary size by 87% compared to existing work with BPE subword tokenization that leads to decreasing memory footprint. Although the size of vocabulary was significantly smaller to the word tokenization, the models with maximum likelihood and unlikelihood training still managed to obtain results across the metrics.

Finally, we need further study the effect of other subword tokenization methods as well as the minimum amount of vocabulary size needed. Furthermore, we need to add a metric based on human evaluations to measure generation quality of subword tokenization in unlikelihood training.

Acknowledgment

This work was supported by Dongseo University, "Dongseo Cluster Project" Research Fund of 2020 (DSU-20200008).

References

[1] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi, The Curious Case of Neural Text Degeneration, in Proc. of International Conference on Learning Representations, 2020.
[2] Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston, Neural Text Generation with Unlikelihood Training, In Proc. of International Conference on Learning Representations, 2020.
[3] OpenAI, Language models are unsupervised multitask learners. *https://openai.com/blog/better-language-models/*
[4] Alexander M. Rush, Yin-Wen Chang, and Michael Collins, Optimal Beam Search for Machine Translation, in Proc. of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 210–221, Oct. 18–21, 2013.
[5] Angela Fan, Mike Lewis, and Yann Dauphin, Hierarchical Neural Story Generation, in Proc. of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers), pp. 889–898, July 15–20, 2018. DOI: https://doi.org/10.18653/v1/P18-1082
[6] Liang Huang, Kai Zhao, and Mingbo Ma, When to Finish? Optimal Beam Search for Neural Text Generation (modulo beam size), in Proc. of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 2134–2139, September 7–11, 2017. DOI: https://doi.org/10.18653/v1/D17-1227
[7] Rico Sennrich, Barry Haddow, Alexandra Birch, Neural Machine Translation of Rare Words with Subword Units, in Proc. of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1715–1725, August, 2016. DOI: https://doi.org/10.18653/v1/P16-1162
[8] Taku Kudo, Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates, in Proc. of 56th Annual Meeting of the Association for Computational Linguistics (Long Papers), pp. 66–75, July 15–20, 2018. DOI: https://doi.org/10.18653/v1/P18-1007
[9] Rohan Chitnis, John DeNero, Variable-Length Word Encodings for Neural Translation Models," in Proc. of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 2088–2093, September 17–21, 2015. DOI: https://doi.org/10.18653/v1/D15-1249
[10] Philip Gage, A New Algorithm for Data Compression, C users Journal, Vol. 12, No. 2, pp. 23-382, June 1994. DOI: https://dl.acm.org/doi/10.5555/177910.177914
[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, Attention is all you need, in Proc. of the 2017 Advances in Neural Information Processing Systems, pages 5998–6008, 2017. DOI: https://dl.acm.org/doi/10.5555/3295222.3295349
[12] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher, Pointer sentinel mixture models, in Proc. of International Conference on Learning Representations, 2017.