INTERSPEECH 2021
30 August – 3 September, 2021, Brno, Czechia

Regularizing Word Segmentation by Creating Misspellings

Hainan Xu, Kartik Audhkhasi, Yinghui Huang, Jesse Emond, Bhuvana Ramabhadran

Google Inc, New York, NY, U.S.A.
{hainan,kaudhkhasi,huangyinghui,emond,bhuv}@google.com

Abstract

This work focuses on improving subword segmentation algorithms for end-to-end speech recognition models, and makes two major contributions. Firstly, we propose a novel word segmentation algorithm. The algorithm uses the same vocabulary generated by a regular wordpiece model, is easily extensible and supports a variety of regularization techniques in the segmentation space, and outperforms the regular wordpiece model. Secondly, we propose a number of novel regularization methods that introduce randomness into the tokenization algorithm, which bring further improvements in speech recognition accuracy, with relative gains up to 8.4% compared to the original wordpiece model. We analyze the methods and show that our proposed methods are equivalent to a sophisticated form of label smoothing, which performs smoothing based on the prefix structures of subword units. A noteworthy discovery from this work is that creating artificial misspellings in words results in the best performance among all the methods, which could inspire future research for strategies in this area.

Index Terms: speech recognition, text representation, subword segmentation, wordpiece model

1. Introduction

Speech recognition is the task of extracting the text information from an audio of speech, which is an integral part of voice assistants like the Google Assistant, and also powers voice typing, which is a more convenient way for people to interact with mobile devices. After decades of research, speech recognition technology has seen dramatic improvements and nowadays these technologies are gradually becoming part of a lot of people's daily lives. The core models for speech recognition have experienced a number of major shifts. With traditional hybrid models [1], a speech recognition system comprises of an acoustic model, a language model, a decoder and a lexicon. In this approach, the lexicon acts as the bridge between the acoustic symbols to word symbols, and therefore researchers need not worry about how text is represented as long as an accurate pronunciation dictionary exists. In the last few years, we have seen the success of end-to-end models [2, 3, 4], which use a single neural network model to directly take the audio input and output text. Such models usually do not have a lexicon to bridge between acoustic features and output units. Therefore, text representation becomes a part of model design, which could potentially impact the model performance.

With end-to-end sequence models, word-based and grapheme-based representations of text are two straightforward methods for representing text. In word-based representation, each word in the vocabulary directly corresponds to a dimension of the input embedding matrix of the label encoder and output embedding matrix of the decoder. With word-based representations, text sequences have shorter lengths compared with other methods, and this makes it easier for models to learn dependency patterns from data; however, this approach needs a pre-defined vocabulary, and cannot handle out-of-vocabulary (OOV) words, and also cannot learn the relation between linguistically related words, e.g. "catch" vs "catching", "book" vs "books", "help" vs "helpful" etc, unless additional information (e.g. letter-based features [5]) is incorporated into the generation of word embeddings. With grapheme-based representation [6], all words are broken into sequences of single characters, and this approach could theoretically learn the relation between similarly spelled words, and accommodate for any OOVs. However, with this approach text sequences are much longer than other representations, which means larger memory footprint and computational time, and also it is relatively harder for models to learn the patterns from data, so second-pass rescoring [7] with an external language model is usually required in order to achieve good performance. Subword-based text representation [8] achieves a trade-off between the two representations above. In this representation, words are broken into subword units using a subword segmentation algorithm. This work focuses on improving the subword-based representation of text, and makes the following contributions:

1. We propose a new word tokenization algorithm, which is highly extensible and achieves better performance compared to the *wordpiece model* [9].
2. We propose a number of regularization methods on top of the new tokenization method, which further improves the performance and outperforms alternative regularization methods like BPE-dropout [10]. We discover that better regularization could be achieved by altering the spelling of words before performing subword segmentation. We observe gains across a variety of languages and the approach is architecture agnostic.
3. We show that our proposed regularization methods are essentially a sophisticated form of label smoothing [11], and this provides a new perspective in analyzing regularization methods in subword algorithms.

2. Prior Work

There has been a lot of work in the space of subword methods for sequence modeling, and in particular, for speech recognition. In [12], Byte-Pair Encoding (BPE) algorithm [13], which was initially proposed for data compression, is used to generate word segmentation for machine translation. BPE iteratively merges pairs of units that have the highest count in data in the training algorithm. In [3], a similar model called wordpiece model is used, which differs from BPE in that the wordpiece model uses language model likelihood instead of pair-counts as selection criteria for merging pairs.

Researchers have taken a number of routes to improve the quality of subword modeling. One such direction is to incorporate other information, e.g. in [14], pronunciation information is utilized for generating phonetically-meaningful subword units.

Another direction is to introduce randomness in the word tokenization process. In [15], subword regularization method is proposed which uses an unigram language model method for randomly generating multiple subword sequences for words. [10] proposes BPE-dropout, which randomly "drops" certain merges when applying the BPE algorithm, and shows gains in machine translation tasks when this technique is applied in model training. In [16], the BPE-dropout method is adapted to the wordpiece model in speech recognition tasks.

3. Methods

3.1. Recap on BPE/wordpiece model

Before describing our method, we first briefly review the BPE/wordpiece model algorithm. The algorithm includes a training part, which analyzes the training corpus and generates a set of subword vocabulary and merge rules; and an application part, which breaks an input word into a sequence of characters, and uses the information from the merging rules to iteratively merge units to form subwords.

Algorithm 1: BPE Model Training

```
Input: a training corpus $C$, desired number of units $n$
Output: a set of vocabulary $V$ and a merge vector $M$
1  V = {all single characters in $C$};
2  M = [];
3  while $size(V) < n$ do
4      all-pairs = {all unit pairs in $C$};
5      [i, j] = arg max$_{[p,q]\in\text{all-pairs}}$ count(C, concat(p, q));
6      $V = V \cup$ {concat(i, j)};
7      $M$.append([i, j]);
8      for all adjacent $[p, q] \in C$ s.t. [p, q] == [i, j] do
9          rewrite C by replacing [p, q] with concat(i, j);
10     end
11 end
12 return $V, M$
```

Algorithm 1 illustrates the training procedure for BPE subword models, where in line 5, Count(x, y) means the number of occurrences of y in x. The algorithm starts by building a vocabulary V consisting of all characters[1], iteratively merging two units from the vocabulary that have the most occurrences in the training corpus, and adding the result of merge to the vocabulary. The algorithm also returns a merge vector M, which is used in the application process of the algorithm, where earlier entries in the vector always have priority over later entries.

The wordpiece model (WPM) algorithm is slightly different from BPE at line 5. Instead of selecting the pair that has the highest count in the training data, wordpiece model would, for each of the merge candidates, temporarily apply that merge to the corpus, and train a language model on that text. Then, it picks the merge that produces the highest language model likelihood of the data of all the merge candidates.

The algorithm for applying the BPE/WPM algorithm is shown in Algorithm 2. Aside from the fact that BPE and wordpiece model might produce different merge vectors, the algorithms for applying the merge vector are exactly the same for both of these algorithms - similar to the training algorithm, it first rewrites the input word into a sequence of characters, and iteratively merges the pair with the highest priority, as defined in the merge vector, until no more merges could be found.

[1]We add a special "begin-of-word"symbol to all words to represent word boundaries but we omit that in our description for simplicity.

Algorithm 2: BPE Model Application

```
Input: merge vector $M$, an input word $w$
Output: subword segmentation of $w$
1 ans = list of characters in $w$;
2 while true do
3     idx = find $i$ such that [ans[i], ans[i + 1]] is in $M$
        with the smallest index;
4     if nothing found then
5         return ans;
6     else
7         ans = merge ans[idx], ans[idx + 1] into one unit
            while keeping everything else unchanged;
8     end
9 end
```

3.2. Proposed Lightweight Wordpiece Model

Our proposed algorithm, which we call a "lightweight wordpiece model", improves upon the application algorithm only, and is independent of the training algorithms used to generate the subword vocabulary. The difference is that we do not use merge-based operations, but only use the trained vocabulary file during the application procedure. The algorithm is shown in Algorithm 3. This algorithm uses a simple left-to-right greedy search for the longest subword present in the vocabulary, until it reaches the end of the word.

Algorithm 3: Lightweight WPM Application

```
Input: vocabulary $V$, an input word $w$
Output: subword segmentation of $w$
1 ans = [];
2 i = 0;
3 while $i < len(w)$ do
4     sub = longest subword of $w$ starting at index $i$ that
        exists in $V$;
5     i = i + len(sub);
6     ans.append(sub);
7 end
8 return ans;
```

3.3. Regularization Methods on top of Lightweight WPM

We also propose two families of regularization methods by introducing randomness to the lightweight WPM algorithm. Adding randomness in the subword segmentation has been shown to help performance in [15, 10], and we will have a further discussion on why it helps in Section 5.2.

3.3.1. Random sampling of subword candidates

In this family of methods, we add randomness in the step that makes the subword selection, in line 4 of Algorithm 3. Instead of always picking the longest subword, we sample from all possible subwords. We propose using the following method to generate the sampling PDF. For a specified rate p, we assign $1 - p$ probability to the longest candidate, and divide the rest p probability evenly among all possible candidates (including the longest). We refer to this method as “uniform smoothed”, and represent this method in our experiment section as “uniform”.

3.3.2. Artificially introducing misspellings

In this family of methods, before applying Algorithm 3, we first introduce some noise into the spelling of words to create misspellings. We use the following two methods to create misspellings.[2]

- “skip”: we randomly delete each character in the input word independently with a predefined probability. For example, “interspeech” could become “intrspeech”.
- “swap”: for each adjacent character pair in the word, we swap their order with a predefined probability. Note, a letter may be swapped at most once – say for the word “the”, if we swap t and h to make it “hte”, then we do not consider swapping the pair “te” after the first swap.

4. Experiments

We report experiment results on a variety of model architectures (RNNT, transformer and conformer) and on different datasets (Librispeech + Voicesearch datasets of a number of languages). By RNNT, we refer to an end-to-end model with LSTMs as both the encoder and decoder trained with RNN-T [17] loss; by transformer, we refer to Google’s Transformer-transducer model described in [18], which uses a transformer as the encoder, an LSTM as decoder, and RNN-T as the training loss function; and conformer refers to a similar model as transformer, but uses a conformer [19] as the acoustic encoder. All those experiments use a subword vocabulary of size 4096 trained per language. We apply various tokenization methods during model training only, and perform standard decoding for producing results. We intend to show two sets of comparisons from each experiment group. First, we compare our lightweight WPM with the regular wordpiece model; also, we compare our regularization methods on top of lightweight WPM against the model without regularization, and against BPE-dropout [10], which also has proven helpful for improving ASR performance [16]. Unless specified otherwise, all regularized models use 0.05 as the BPE-dropout/skip/swap/uniform regularization rate.

4.1. Results on Librispeech (English)

In this section, we present Librispeech [20] results with a Transformer model. The training is done on all 960 hours of Librispeech training data, and Spec-augment [21] is used during training. We report WER results on test-clean and test-other set, with different tokenizers. In our experiments, each self-attention layer has access to full utterance as context, and no external language model is used [3]. We can see, 1) the lightweight WPM slightly outperforms the regular WPM; 2) all our regularization brings significant gains over the baseline, and either on-par with or outperforms BPE-dropout.

[2]In both methods, we treat the “begin-of-word” symbol as a normal character, which can be skipped/swapped. To avoid confusion, we do not include the special character in our examples or in our later analysis.

[3]We point out that state-of-the-art performance on Librispeech are 1.9% and 3.9% on test-clean and test-other, albeit with unsupervised transcription of additional training audio, fusion with external LMs and use of different architectures [22].

Table 1: *Librispeech 960h Transformer Model Results with Different Wordpiece Algorithms*

tokenizer	test-clean	test-other
WPM	2.9	6.3
+ BPE-dropout	2.4	5.8
lightweight	2.8	6.0
+ uniform	2.4	5.7
+ swap	2.5	5.7
+ skip	2.4	5.6

4.2. Results for Other Languages

In this section, we report WER performance of models for a number of languages, on Google’s Voicesearch datasets. Readers are referred to [23] for detailed descriptions of the data. During training, domain ID information represented as a one-hot vector is appended to the input features to make the models domain-aware, and all model encoders only uses left context in the computation, so the models support streaming operations.

Table 2 reports Telugu and Urdu Voicesearch results, both with RNNT and conformer models. All those models are initialized with a trained Indian English model. We see that overall, the lightweight WPM brings a slight improvement over the regular wordpiece model (WPM), and all of our proposed regularizations bring gains over the baseline, regardless of architecture; the “skip” and “swap” methods can consistently outperform BPE-dropout methods. We will present more discussion on why those methods work in Section 5.2

Table 2: *Telugu and Urdu Results on Voicesearch*

	RNNT		conformer	
tokenizer	Telugu	Urdu	Telugu	Urdu
WPM	25.0	17.4	24.5	16.7
+ BPE-dropout	24.6	16.3	24.0	16.3
lightweight	24.7	17.0	24.4	16.3
+ uniform	25.1	16.2	24.2	15.6
+ swap	24.5	16.4	**23.8**	15.4
+ skip	**24.3**	**16.0**	23.9	**15.3**
best rel. gain	-2.8%	-8.0%	-2.9%	-8.4%

Table 3: *Gujarati and Kanada Results on Voicesearch*

tokenizer	Gujarati (conformer)	Kanada (RNNT)
WPM	20.3	30.2
lightweight	20.1	30.3
+ skip	19.8	**28.6**
+ swap	**19.7**	28.7

In Tables 3, 4, we present Voicesearch results on more Indic languages (Gujarati and Kannada), and Swedish, a Nordic language. We compare the lightweight WPM, with or without regularization techniques, against the regular wordpiece model. In all those models, we see that (1) the lightweight WPM consistently matches or beats the regular WPM, and (2) the “swap” and “skip” regularization consistently brings significant gains, with “skip” having a slight edge overall.

Table 4: *Swedish RNNT Results on Voicesearch*

tokenizer	Voicesearch
WPM	13.2
+ BPE-dropout	12.7
lightweight	13.2
+ swap	**12.2**
+ skip	**12.2**

Table 5: *Librispeech 960h Results Comparing Different Versions of "skip" Regularization on Different Base Algorithms*

tokenizer	unit to skip	test-clean	test-other
WPM	N/A	2.9	6.3
WPM	letters	3.0	6.1
lightweight	letters	2.4	5.6
lightweight	subwords	2.6	6.0
grapheme	N/A	2.8	6.2
grapheme	letters	2.5	6.0

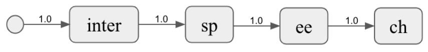

Figure 1: *Gold Reference in Training without Regularization*

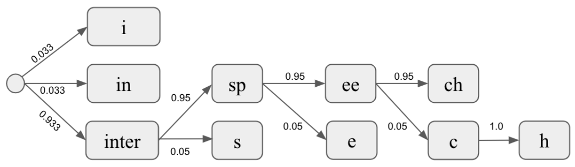

Figure 2: *Gold Reference with "uniform" Regularization*

5. Analysis

5.1. Creating misspellings for regular WPM methods

Readers might be interested in whether it is effective to combine the method of creating artificial misspellings with the original wordpiece model. We conducted such experiments applying the "skip" regularization to regular WPM algorithm on the Librispeech dataset and report results in the top 2 rows of Table 5. The numbers show that the "skip" operation overall brings no gains in ASR performance, and that the "skip" method only works with the lightweight wordpiece model.

5.2. Connection between Wordpiece Regularization and Label Smoothing

We point out that our "uniform" method is essentially a special form of label smoothing [4, 24], that assigns probability masses to subwords that are prefixes of the gold label subword [4]. We refer to this as the *prefix structure* of a subword vocabulary. We illustrate this point with the following example.

Figure 1 shows what happens when we train our ASR models on the word "interspeech" represented with the regular wordpiece model. In this Figure, the circle represents beginning of the word, an arrow points to a subword that is used as the gold for the current position, and the number on the arrow indicates the probability of the corresponding gold label. Note, in this case, the probability is always 1.0 since the tokenization is deterministic and exactly one subword is used as gold at any position.

Figure 2 shows a partial graph of the training procedure for the same word "interspeech", with "uniform" regularization rate = 0.1 (meaning we first assign probability 0.9 for the longest candidate, and divide the remaining 0.1 evenly among *all* candidates, including the longest). We see that, at different positions, multiple subword candidates could be sampled that share the same prefix. This will have two effects: (1) it makes the gold distribution less sharp and therefore prevents over-confidence of the model and (2) it makes the model aware of the prefix relations of the subwords, instead of in the case of regular WPM, where all subword units are treated as independent symbols.

[4]Note, label smoothing is usually applied to frame-level loss functions, e.g. cross-entropy; for a sequence-level loss like RNN-T, it is highly non-trivial.

In the case of methods that alter the spelling, e.g. with "skip", since any letter could potentially be skipped, then all of the subword candidates shown in Figure 2 will also show up in the "skip" case; additionally, the "skip" model could learn additional relations beyond prefixes, since any similarity of spelling in general could be revealed with "skip" operation. For example, if the gold label is "ing", and letter "i" is skipped, then the model will assign probability to the subword "ng", and thus learn the relation between "ing" and "ng". Furthermore, since multiple letters could be skipped, the gold label at any position could be from further into the future, and this could help model learn better long-term dependencies in the data.

5.3. Importance of Prefix Structure in Label Smoothing for ASR Performance

We conduct another set of experiments, which skips whole token units during the tokenization – skipping subwords with a lightweight WPM model and skipping letters with a grapheme model. This helps us to see the specific impact of the prefix structure between different subword units. We present our results in Table 5 (the bottom 4 rows). From those numbers, we see that in the case of skip for grapheme, and skip subwords for the lightweight WPM, a smaller gain is seen than performing skip letter for lightweight WPM, proving that taking advantage of the prefix relations between subwords plays an important part in the gains brought by our proposed methods.

6. Conclusion

In this paper, we propose a new subword segmentation algorithm which we refer to as a lightweight WPM, which uses a greedy left-to-right search that always picks the longest subword from the word. The algorithm is highly extensible, and we propose a number of regularization techniques, both by sampling from multiple subword candidates, and also by altering the spelling of the words before the subword segmentation procedure. Experiments show that the lightweight WPM achieves overall superior performance compared to the regular WPM, and our proposed regularization methods significantly improve performance, and outperform BPE-dropout. We hope that this research opens up explorations in other subword-based regularization methods, and in other language families as well, including non-alphabetic languages.

7. References

[1] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. CONF. IEEE Signal Processing Society, 2011.

[2] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, "Listen, attend and spell," *arXiv preprint arXiv:1508.01211*, 2015.

[3] C.-C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina *et al.*, "State-of-the-art speech recognition with sequence-to-sequence models," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4774–4778.

[4] Y. Wang, T. Chen, H. Xu, S. Ding, H. Lv, Y. Shao, N. Peng, L. Xie, S. Watanabe, and S. Khudanpur, "Espresso: A fast end-to-end neural speech recognition toolkit," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019, pp. 136–143.

[5] H. Xu, K. Li, Y. Wang, J. Wang, S. Kang, X. Chen, D. Povey, and S. Khudanpur, "Neural network language modeling with letter-based features and importance sampling," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2018, pp. 6109–6113.

[6] M. Killer, S. Stuker, and T. Schultz, "Grapheme based speech recognition," in *Eighth European Conference on Speech Communication and Technology*, 2003.

[7] H. Xu, T. Chen, D. Gao, Y. Wang, K. Li, N. Goel, Y. Carmiel, D. Povey, and S. Khudanpur, "A pruned rnnlm lattice-rescoring algorithm for automatic speech recognition," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2018, pp. 5929–5933.

[8] M. Labeau and A. Allauzen, "Character and subword-based word representation for neural language modeling prediction," in *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, 2017, pp. 1–13.

[9] M. Schuster and K. Nakajima, "Japanese and korean voice search," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 5149–5152.

[10] I. Provilkov, D. Emelianenko, and E. Voita, "Bpe-dropout: Simple and effective subword regularization," *arXiv preprint arXiv:1910.13267*, 2019.

[11] R. Müller, S. Kornblith, and G. Hinton, "When does label smoothing help?" *arXiv preprint arXiv:1906.02629*, 2019.

[12] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," *arXiv preprint arXiv:1508.07909*, 2015.

[13] P. Gage, "A new algorithm for data compression," *C Users Journal*, vol. 12, no. 2, pp. 23–38, 1994.

[14] H. Xu, S. Ding, and S. Watanabe, "Improving end-to-end speech recognition with pronunciation-assisted sub-word modeling," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 7110–7114.

[15] T. Kudo, "Subword regularization: Improving neural network translation models with multiple subword candidates," *arXiv preprint arXiv:1804.10959*, 2018.

[16] H. Xu, Y. Huang, Y. Zhu, K. Audhkhasi, and B. Ramabhadran, "Convolutional dropout and wordpiece augmentation for end-to-end speech recognition," in *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021.

[17] A. Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711*, 2012.

[18] Q. Zhang, H. Lu, H. Sak, A. Tripathi, E. McDermott, S. Koo, and S. Kumar, "Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7829–7833.

[19] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," *arXiv preprint arXiv:2005.08100*, 2020.

[20] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[21] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[22] D. S. Park, Y. Zhang, Y. Jia, W. Han, C.-C. Chiu, B. Li, Y. Wu, and Q. V. Le, "Improved noisy student training for automatic speech recognition," *arXiv preprint arXiv:2005.09629*, 2020.

[23] A. Kannan, A. Datta, T. N. Sainath, E. Weinstein, B. Ramabhadran, Y. Wu, A. Bapna, Z. Chen, and S. Lee, "Large-scale multilingual speech recognition with a streaming end-to-end model," *arXiv preprint arXiv:1909.05330*, 2019.

[24] G. Pereyra, G. Tucker, J. Chorowski, Ł. Kaiser, and G. Hinton, "Regularizing neural networks by penalizing confident output distributions," *arXiv preprint arXiv:1701.06548*, 2017.