SUBWORD REGULARIZATION AND BEAM SEARCH DECODING FOR END-TO-END AUTOMATIC SPEECH RECOGNITION

Jennifer Drexler and James Glass

MIT Computer Science and Artificial Intelligence Laboratory
Cambridge, MA, USA
{jdrexler, glass}@mit.edu

ABSTRACT

In this paper, we experiment with the recently introduced subword regularization technique [1] in the context of end-to-end automatic speech recognition (ASR). We present results from both attention-based and CTC-based ASR systems on two common benchmark datasets, the 80 hour Wall Street Journal corpus and 1,000 hour Librispeech corpus. We also introduce a novel subword beam search decoding algorithm that significantly improves the final performance of the CTC-based systems. Overall, we find that subword regularization improves the performance of both types of ASR systems, with the regularized attention-based model performing best overall.

***Index Terms*—** automatic speech recognition, subword units, beam search, CTC, attention

1. INTRODUCTION

Within the field of automatic speech recognition (ASR), determining the correct scale of units to use at various stages in the recognition pipeline is a key research area. The idea of finding and using a vocabulary of "subword units" - sequences of one-or-more phonemes or characters - has been explored extensively in the context of traditional HMM-based state-dependent acoustic models [2, 3] and has become increasingly common at the output of end-to-end deep neural network models, such as connectionist temporal classification (CTC) and attention-based models [4, 5, 6]. For recent end-to-end ASR models, subword units are most often discovered using the byte pair encoding (BPE) technique [7], which was originally developed for machine translation.

Subword regularization [1] is a more recent technique for both discovering and using subword units that has been shown to produce large gains (sometimes several BLEU points) over high-quality machine translation baselines that use BPE. Rather than deterministically splitting every word into subword units, subword regularization involves learning a probabilistic model over subword units and sampling a new segmentation for every word each time it appears.

In this paper, we apply subword regularization to ASR, where it has not previously been tested. We focus on end-to-end ASR, exploring both CTC-based models and attention-based models. While the original subword regularization experiments on machine translation suggest its general usefulness with attention-based models, this paper is the first attempt to combine subword regularization with CTC.

In addition to these experiments, we develop a modified beam search algorithm to improve the performance of CTC models with subword regularization. Our subword beam search algorithm combines the likelihoods of different segmentations of the same word during decoding, thus finding the most likely word sequence rather than the most likely sequence of subword units. We show that the performance of CTC models drops drastically with increased subword regularization when standard beam search is used, but that our algorithm completely erases these losses.

2. SUBWORD REGULARIZATION

Subword regularization [1] is based on a simple unigram language model (LM). Given a vocabulary of subword units, such a model is easy to train, and Viterbi search can efficiently find the best segmentation for any word according to that model. Alternatively, segmentations can be sampled from the language model. Kudo [1] introduced a technique for joint learning of the unigram LM and a vocabulary of a desired size. The vocabulary always includes all of the single characters in the alphabet, so that there will never be out-of-vocabulary words. This technique starts with the set of the most frequent strings in the corpus, a set much larger than the desired vocabulary size. Given this seed vocabulary, the LM is computed. Kudo then computes the "loss" associated with each unit - the reduction in the overall likelihood of the corpus if that unit were to be left out. The LM is then re-estimated using the 80% of subword units with the highest associated loss. This process is repeated until the appropriate vocabulary size is reached.

Once the LM and vocabulary are fixed, we can sample a segmentation from the following multinomial distribution:

$$P(x_i|X) \cong \frac{p(x_i)^\alpha}{\sum_{j=1}^{n} p(x_j)^\alpha}$$

978-1-5386-4658-8/18/$31.00 ©2019 IEEE

where n is the number of n-best segmentations used to approximate the true distribution and α is a smoothing parameter. $\alpha = 0$ creates a uniform distribution, while larger settings of α move closer to the Viterbi segmentation. Kudo uses the forward-filtering and backward-sampling algorithm [8] for exact sampling from all possible segmentations. All experimental results in this paper use this setting.

We make some minor modifications to the technique described in [1]: in addition to specifying a desired vocabulary size, we specify the maximum subword length in characters, which is easily implemented by only considering subwords up to that length in the seed vocabulary. In [1], subwords do not cross word boundaries, but each space is included as part of the following word. We treat all spaces as stand-alone characters, not included in any subword units.

3. SUBWORD BEAM SEARCH DECODING

Prefix beam search decoding is a beam search algorithm specifically designed for use with CTC models [9]. It is a dynamic programming algorithm in which we step through time, keeping the n prefixes with the highest cumulative probability at each step. The probability of outputting a prefix $\boldsymbol{p}$ by time t given input sequence x is:

$$p(\boldsymbol{p}|x,t) = \gamma(\boldsymbol{p}_b, t) + \gamma(\boldsymbol{p}_n, t)$$

where $\gamma(\boldsymbol{p}_b, t)$ is the probability of outputting prefix $\boldsymbol{p}$ by time t such that the blank label is output at time t. Similarly, $\gamma(\boldsymbol{p}_n, t)$ is the probability of outputting prefix $\boldsymbol{p}$ by time t such that a non-blank label is output at time t.

In our updated decoding algorithm, which we call subword prefix beam search decoding, we add an extra parameter to our definition of γ, such that $\gamma(\boldsymbol{p}_n, z, t)$ is the probability of outputting prefix $\boldsymbol{p}$ by time t such that a non-blank label *of length* z is output at time t. With these new γ definitions, we then redefine our prefix probability as:

$$p(\boldsymbol{p}|x,t) = \gamma(\boldsymbol{p}_b, t) + \sum_{z=1}^{M} \gamma(\boldsymbol{p}_n, z, t)$$

where M is the maximum subword unit length in our vocabulary. Note that this algorithm reduces to the original beam search algorithm when all units are single characters. The updated algorithm is shown in Algorithm 1.

On line 10 of Algorithm 1, to find out if $\boldsymbol{p}$ ends in the label k, we need to maintain a mapping from each label k to its constituent characters. We can then check if the last z characters in $\boldsymbol{p}$ match the characters that make up k.

In standard prefix beam search, the γ definitions match the forward variables in the forward-backward CTC training algorithm, described in detail in [9]. Our updated γ variables match the forward variables in the modified forward-backward training algorithm used in Gram CTC [10], described in more detail in the following section.

Algorithm 1 Subword Prefix Beam Search Decoding

```
 1: $P = \{\}$
 2: for t=1 to T do
 3:     for all labels $k \in A$ do
 4:         $z = labelLength[k]$
 5:         for all prefixes $\boldsymbol{p} \in P$ do
 6:             $\boldsymbol{p}^* = \boldsymbol{p} + k$
 7:             $P.add(\boldsymbol{p}^*)$
 8:             if $k == blank$ then
 9:                 $\gamma(\boldsymbol{p}_b, t) \mathrel{+}= y_k^t * p(\boldsymbol{p}, t-1)$
10:             else
11:                 if $\boldsymbol{p}$ ends in $k$ then
12:                     $\gamma(\boldsymbol{p}_n, z, t) \mathrel{+}= y_k^t * \gamma(\boldsymbol{p}_n, z, t-1)$
13:                     $\gamma(\boldsymbol{p}_n^*, z, t) \mathrel{+}= y_k^t * [p(\boldsymbol{p}, t-1) - \gamma(\boldsymbol{p}_n, z, t-1)]$
14:                 else
15:                     $\gamma(\boldsymbol{p}_n^*, z, t) \mathrel{+}= y_k^t * p(\boldsymbol{p}, t-1)$
16:                 end if
17:             end if
18:         end for
19:     end for
20:     $sortedP = sort(P)$
21:     $P = topNPrefixes(sortedP, n)$
22: end for
```

4. PRIOR WORK

The idea of finding and using a vocabulary of subword units is a longstanding one within ASR. These units were originally necessary to avoid the out-of-vocabulary (OOV) problem with traditional acoustic and pronunciation models [2, 11, 3]. More recent character-based end-to-end models [12] do not have an OOV problem, but researchers have still found advantages to using a larger vocabulary of subword units as opposed to characters.

The subword units used with end-to-end ASR models are typically discovered using byte pair encoding (BPE) [7], which learns a subword vocabulary from a text corpus that can be used to deterministially segment any word. BPE units have been shown to improve ASR performance when used with attention-based systems [5, 13], CTC-based systems [4], and hybrid attention-CTC models [6]. This prior work shows a consistent trend of small improvements due to subword units, with the best performance coming with relatively small vocabularies of 300 or 500 units. These papers also show more improvement to attention-based and hybrid models than to CTC models.

An alternative method of training CTC-based ASR models with subword units, called "Gram CTC," was introduced in [10]. During Gram CTC training, the CTC loss marginalizes over all possible correct transcripts formed by using an combination of subword units. For example, the word "OF" could be produced either by individually outputting the characters "O" and "F" or by outputting the unit "OF". This is closely related to our decoding method, which similarly considers all possible segmentations of the same word sequence.

Liu et al. [10] also use Gram CTC as a subword unit discovery method, starting with all possible subwords up to a given length and then iteratively selecting the units most often used by their trained model during greedy decoding.

5. DATA AND METHODS

5.1. Data

We did experiments using the Wall Street Journal Corpus[14] and the Librispeech[1] corpus. For WSJ, we used the full SI284 training set (80 hours), the dev93 set for validation, and eval92 for testing. All results are on eval92. We normalize all text to contain only alphanumeric characters, along with a 'SPACE' symbol. For Librispeech, we used the full 1,000h training set, combined the clean and "other" dev sets for validation, and report separate results on the clean and other test sets. Word-level n-gram language models for both datasets were trained on the accompanying text corpora; we use a 3-gram LM for librispeech and a 4-gram LM for WSJ.

5.2. ASR Models

5.2.1. Attention-based Model

Our attention-based ASR model architecture comes from the Listen, Attend, and Spell model [12]. It's an encoder-decoder model with an MLP attention mechanism [15]. The encoder has four layers: the first is a bi-directional LSTM layer, the three subsequent layers are pyramidal bLSTM layers [12], each of which downsamples its input sequence by a factor of two. The decoder has two LSTM layers. All recurrent layers have 256 units each; the embedding layer in the decoder has 64 units. During training, we use a fixed sampling rate of 10% in the decoder [12].

This model is trained end-to-end to maximize the log likelihood of the correct label sequence during training. We use stochastic gradient descent (SGD) with momentum for optimization. During inference, we use beam search decoding [16]. We perform beam search with a beam size of five on the validation set to determine the best performing epoch, then report results of decoding the test set with beam size 20, if not otherwise specified. Decoding with an n-gram language model was implemented using WFSTs, following [17]. We use a language model weight of 0.5 and character-level bonus of 1.0 to encourage longer outputs.

5.2.2. CTC-based Model

For CTC-based ASR, we use a variant of the DeepSpeech2 model [18]. This model stacks several different types of neural network layers - two convolutional layers with tanh nonlinearities, followed by five GRU layers of 800 units each, a fully-connected layer, and a softmax layer - all of which are trained jointly with the CTC loss function [19].

Segmentation	α	clean WER (+ LM)	other WER (+ LM)
Character		11.9 (8.3)	31.1 (24.4)
Unigram, 200, ≤ 3	∞	11.9 (8.1)	30.5 (23.1)
	2	12.3 (**7.4**)	30.4 (22.0)
	1	12.4 (7.7)	30.2 (22.5)
	0.5	13.8 (8.9)	31.8 (24.6)
Unigram, 500, ≤ 4	∞	**11.7** (8.2)	29.9 (23.0)
	2	12.6 (7.8)	29.9 (**21.7**)
	1	12.1 (8.0)	**29.4** (22.4)
	0.5	12.4 (9.7)	30.7 (25.3)

Table 1. Results from the CTC model on the Librispeech dataset. α is the regularization parameter for segmentation sampling; $\alpha = \infty$ denotes the Viterbi segmentation.

We again use SGD with momentum for optimization and use the same procedure as above for validation and testing. When decoding with a language model, we use a language model weight of 0.8 and word-level bonus of 1.0.

5.3. Software

Our attention-based ASR model was built with the OpenNMT toolkit[2], with some minor modifications. The CTC model used the deepspeech.pytorch[3] codebase, which uses ctcdecode[4] for beam search decoding[5]. Subword regularization was performed with Google's sentencepiece tokenizer[6].

6. RESULTS AND ANALYSIS

Table 1 shows the results of subword regularization with the CTC model on the Librispeech dataset. α is the regularization parameter; $\alpha = \infty$ means we sampled the deterministic best segmentation for each word. All results shown used the subword prefix beam search algorithm, which is identical to regular beam search in the character segmentation case and was consistently the same or slightly better than regular beam search on the unregularized subword models. Overall, using subwords units is as good as or better than using characters, and subwords have a bigger impact on the "other" test set. Without an LM, subword regularization gives additional improvement on the "other" test set but worsens performance on the clean test set. This result are in line with the finding in [1] that subword regularization improves model robustness on out-of-domain test sets. In combination with an LM, subword regularization improves performance on both test sets.

[1] http://www.openslr.org/12/

[2] https://github.com/OpenNMT/OpenNMT-py/

[3] https://github.com/SeanNaren/deepspeech.pytorch

[4] https://github.com/parlance/ctcdecode

[5] Updated code available at https://github.com/jdrex/ctcdecode

[6] https://github.com/google/sentencepiece

Segmentation	α	WER (no LM)	sWER (no LM)	 (+ LM)
Character		19.8	19.8	16.1
Unigram, 100, ≤ 2	∞	20.0	20.0	15.1
	10	19.8	19.5	14.1
	5	19.4	**18.8**	**14.0**
	2	22.0	19.5	14.8
	1	28.5	20.6	15.5
	0.5	37.9	22.0	15.7

Table 2. Results from the CTC model on the WSJ dataset. WER denotes results using the standard prefix beam search algorithm; sWER results use our updated algorithm.

Segmentation	α	WER No LM	 + LM
Character		16.0	12.4
Unigram, 100 units, ≤ 2	∞	16.0	12.1
	1	14.1	**10.7**
	0.5	14.2	11.6
	0.2	14.3	11.5
Unigram, 200 units, ≤ 4	∞	15.1	11.8
	1	**14.0**	**10.7**
	0.5	14.3	11.1
	0.2	14.8	11.0

Table 3. Results from the encoder-decoder model with attention on the WSJ dataset.

Table 2 has results of subword regularization of the CTC model on the WSJ dataset. Here, we show WER using both the standard prefix beam search algorithm (“WER”) and our subword prefix beam search algorithm (“sWER”). The lower the regularization parameter, the worse the standard beam search algorithm performs, and thus the bigger the impact of our modified algorithm. CTC is generally known to produce highly “peaked” output distributions, but as we further regularize our model by lowering the α parameter, our model becomes “torn” between multiple possible options at some timesteps. The outputs are still quite peaked, with most of the probability mass placed on the two or three most likely outputs per frame, but this change is enough to cause a large degradation in beam search performance.

Both the problem created by these less peaked distributions and the solution provided by our beam search algorithm can be demonstrated with a simple example of one of the most frequently misspelled words in the standard beam search output of the highly regularized models: the word “FOR.” This word is frequently covered by the outputs at two subsequent frames, the first of which is undecided between outputting “F” and “FO” while the second is undecided between “R” and “OR”. In the standard beam search case, there are four possible, distinct paths across these two frames: F-R, FO-R, F-OR, and FO-OR. If the model is almost evenly split between its two choices on both frames, these four paths each have close to the same likelihood. Frequently, F-R or FO-OR is the most likely combination, leading to an output error. Subword prefix beam search, however, collapses FO-R and F-OR into a single hypothesis (as both produce the word FOR), and the word is thus always recognized correctly.

As in Table 1, Table 2 shows improvement due to subword regularization both with and without an LM. In this case, we see a larger improvement (both absolute and relative) to the subword-based models than the character-based baseline from the addition of the LM.

Table 3 shows the results of subword regularization with the attention-based model on the WSJ dataset. Overall, the attention-based model performs better than the CTC-based one on the WSJ dataset. As in previous work, we find that the use of subword units is more impactful with the attention-based model than the CTC model. Additionally, we see that subword regularization produces larger improvements over the Viterbi segmentation when used with the attention-based model than when used with CTC.

We analyzed the attention-based beam search and found that the majority of the beams contained the same final hypothesis. For example, out of 20 beams, there were, on average, only 2.19 unique outputs from our model with 100 units and $\alpha = 0.5$. Unlike with CTC, we cannot collapse multiple beams during decoding because of the recurrent attention-based decoder. Instead, we implemented a simple post-processing where we add up the probabilities of all beams with the same recognition hypothesis and return the overall best hypothesis, but found that this had negligible impact on the results.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we tested subword regularization with both CTC-based and attention-based ASR models. We were consistently able to find a subword setting that out-performed the character baseline. When used with our novel subword prefix beam search decoding algorithm, subword regularization added additional benefit to the CTC model on the WSJ dataset and the other Librispeech test set. The biggest improvement was to the attention-based model, where subword regularization produced significant gains above the unregularized subword model.

In future work, we hope to expand upon the experiments presented here with a direct comparison between Gram CTC and subword regularization. This will show the relative merits of considering multiple segmentations of each word during training and decoding. We are also hopeful that subword prefix beam search decoding will improve the Gram CTC-trained model as it improved the CTC models used here.

8. REFERENCES

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

[2] Issam Bazzi and James Glass, "Heterogeneous lexical units for automatic speech recognition: preliminary investigations," in *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*. IEEE, 2000, vol. 3, pp. 1257–1260.

[3] Ivan Bulyko, José Herrero, Chris Mihelich, and Owen Kimball, "Subword speech recognition for detection of unseen words," in *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[4] Thomas Zenkel, Ramon Sanabria, Florian Metze, and Alex Waibel, "Subword and crossword units for CTC acoustic models," *arXiv preprint arXiv:1712.06855*, 2017.

[5] Albert Zeyer, Kazuki Irie, Ralf Schlüter, and Hermann Ney, "Improved training of end-to-end attention models for speech recognition," *arXiv preprint arXiv:1805.03294*, 2018.

[6] Zhangyu Xiao, Zhijian Ou, Wei Chu, and Hui Lin, "Hybrid CTC-attention based end-to-end speech recognition using subword units," *arXiv preprint arXiv:1807.04978*, 2018.

[7] Rico Sennrich, Barry Haddow, and Alexandra Birch, "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)*, 2016, vol. 1, pp. 1715–1725.

[8] Steven L Scott, "Bayesian methods for hidden Markov models: Recursive computing in the 21st century," *Journal of the American Statistical Association*, vol. 97, no. 457, pp. 337–351, 2002.

[9] Alex Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*, Springer, 2012.

[10] Hairong Liu, Zhenyao Zhu, Xiangang Li, and Sanjeev Satheesh, "Gram-CTC: Automatic unit selection and target decomposition for sequence labelling," in *International Conference on Machine Learning*, 2017, pp. 2188–2197.

[11] Jan Kneissler and Dietrich Klakow, "Speech recognition for huge vocabularies by using optimized subword units," in *Seventh European Conference on Speech Communication and Technology*, 2001.

[12] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 4960–4964.

[13] Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J Weiss, Kanishka Rao, Ekaterina 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.

[14] LDC, "Wall Street Journal Corpus," 1994.

[15] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[16] Ilya Sutskever, Oriol Vinyals, and Quoc V Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[17] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio, "End-to-end attention-based large vocabulary speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 4945–4949.

[18] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al., "Deep speech 2: End-to-end speech recognition in English and Mandarin," in *International Conference on Machine Learning*, 2016, pp. 173–182.

[19] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 369–376.