WORD-LEVEL SPEECH RECOGNITION WITH A DYNAMIC LEXICON

A PREPRINT

Ronan Collobert **Awni Hannun** **Gabriel Synnaeve**
Facebook AI Research
{locronan,awni,gab}@fb.com

June 12, 2019

ABSTRACT

We propose a direct-to-word sequence model with a dynamic lexicon. Our word network constructs word embeddings dynamically from the character level tokens. The word network can be integrated seamlessly with arbitrary sequence models including Connectionist Temporal Classification and encoder-decoder models with attention. Sub-word units are commonly used in speech recognition yet are generated without the use of acoustic context. We show our direct-to-word model can achieve word error rate gains over sub-word level models for speech recognition. Furthermore, we empirically validate that the word-level embeddings we learn contain significant acoustic information, making them more suitable for use in speech recognition. We also show that our direct-to-word approach retains the ability to predict words not seen at training time without any retraining.

1 Introduction

Predicting words directly has the potential to allow for more accurate, more efficient and simpler end-to-end automatic speech recognition (ASR) systems. For example, outputting words enables direct optimization of the word error rate (WER), which more directly optimizes the quality of the transcription. While phonemes last 50-150 milliseconds and graphemes can be shorter or even silent, words are on average much longer. This allows the acoustic model to capture meaning more holistically, operating at a lower frequency with a larger context. To operate at a lower frequency the acoustic model takes larger strides (e.g. sub-samples more), which contributes to a much faster transcription speed and a lower memory footprint.

One major hurdle for direct-to-word approaches is transcribing words not found in the training vocabulary. Unlike phone-based or letter-based systems, the output lexicon in a word-based model is bounded by the words observed in the training transcriptions. Here, we propose an end-to-end model which outputs words directly, yet is still able to dynamically modify the lexicon with words not seen at training time. The method consists in jointly training an acoustic model which outputs word embeddings with a sub-word model (e.g. graphemes or word pieces) which also outputs word embeddings. The transcription of a new word is generated by inputting the individual tokens of the word into the sub-word model to obtain an embedding. We then match this embedding to the output of the acoustic model. For instance, assume the word "caterpillar" was not seen at training time, but "cat" and "pillar" were. If we input "caterpillar" into the word embedding model it should yield an embedding close to the embedding of the acoustic model when observing speech with the words "cat" and "pillar".

Another hurdle for direct-to-word approaches is the need for massive datasets since some words may be seen only a few times in a data set with even 1,000 hours of speech. For example, Soltau et al. [31] demonstrate competitive word-based speech recognition but require 100,000 hours of captioned video to do so. Our approach circumvents this issue by learning an embedding from the sub-word tokens of the word. By jointly optimizing the word model and the acoustic model, we gain the added advantage of learning acoustically relevant word embeddings. In order to efficiently optimize both the acoustic and word models simultaneously, we propose a simple sampling based approach to approximate the normalization term over the full vocabulary. We also show how to efficiently integrate the word model with a beam search decoder.

We validate our method on two commonly used models for end-to-end speech recognition. The first is a Connectionist Temporal Classification (CTC) model [15] and the second is a sequence-to-sequence model with attention (seq2seq) [4, 10, 32]. We show competitive performance with both approaches and demonstrate the advantage of predicting words directly, especially when decoding without the use of an external language model. We also validate that our method enables the use of a dynamic lexicon and in particular can accurately predict words not seen at training time.

2 Related Work

Our work builds on a large body of research in end-to-end sequence models in general and also specifically in speech recognition [1, 5, 9, 11]. The direct-to-word approach can be used with structured loss functions including Connectionist Temporal Classification (CTC) [15] and the Auto Segmentation Criterion (ASG) [11] or less structured sequence-to-sequence models with attention [4].

Unlike lexicon-free approaches [22, 25, 33] our model requires a lexicon. However, the lexicon is adaptable allowing for a different lexicon to be used at inference than that used during training. Traditional speech recognizers also have adaptable lexicons, which they achieve by predicting phonemes and requiring a model for grapheme-to-phoneme conversion [7, 29]. Direct to grapheme speech recognition [18, 19] circumvents the need for the grapheme-to-phoneme model but suffers from other drawbacks including (1) inefficiency due to a high-frame rate and (2) optimizing the letter error rate instead of the word error rate. In contrast, we show that it is possible to predict words directly while retaining an adaptable lexicon.

Prior work has also attempted direct-to-word speech recognition [2, 21, 31]. These approaches require massive data sets to work well [31] and do not have adaptable lexicons. Similar to our work is that of Bengio and Heigold [6], who learn to predict word embeddings by using a sub-word level embedding model. However, their model is not end-to-end in that it requires an aligned training dataset and only uses the word model for lattice re-scoring. They also use a triplet ranking loss to learn the word embeddings, whereas our word model integrates seamlessly with common loss functions such as CTC or categorical cross entropy.

One benefit of the direct-to-word approach is that it more directly optimizes the WER. Prior work has attempted to build structured loss functions which optimize a higher-level error metric. Discriminative loss functions like Minimum Bayes Risk training can be used to optimize the WER directly but are not end-to-end [13]. On the other hand, end-to-end approaches are often non-differentiable [14, 27], or can be quite complex and difficult to implement [12]. In contrast, the word model we propose is easy to use with existing end-to-end sequence models.

Similar approaches to building word representations from characters have been studied for tasks in language processing [8, 23]. Santos and Zadrozny [30] apply a convolutional model to construct word embeddings from characters. Our work differs from this work in that (1) we show how to integrate word-level embeddings with structured loss functions like CTC used in speech recognition and (2) we use a sampling approach to efficiently compute the vocabulary level normalization term when training the model. Ling et al. [24] propose a hierarchical character-to-word based sequence-to-sequence model for machine translation. This approach generates new words character-by-character and hence is more expensive and not easily generalizable for use with structured loss functions like CTC.

3 Model

We consider a standard speech recognition setting, where we are given audio-sentence pairs $(\mathbf{X}, \mathbf{Y})$ at training time. $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_S\}$ is a sequence of acoustic frames (e.g. log-mel filterbanks), and $\mathbf{Y} = \{y_1, \ldots y_N\}$ is a sequence of words (with $y_i \in \mathcal{D}$). We are interested in models mapping $\mathbf{X}$ to $\mathbf{Y}$ in an end-to-end manner. We consider two approaches: (1) a structured-output learning approach, leveraging a sequence-level criterion like CTC, and (2) a sequence-to-sequence (seq2seq) approach, leveraging an encoder-decoder model with an attention mechanism. Figures 1 and 2 show a high-level overview of our proposed approach combined with CTC and seq2seq models respectively.

As mentioned in Section 1, word-level end-to-end approaches face generalization issues regarding rare words, as well as challenges in addressing out-of-vocabulary words (which commonly occur at inference time in ASR). To cope with these issues, we consider an acoustic model $f^{am}(\mathbf{Y}|\mathbf{X}, \mathbf{W})$ which relies on *word embeddings* to represent any word $y \in \mathcal{D}$ in the lexicon by a d-dimensional vector $\mathbf{W}_y \in \mathbb{R}^d$. As the acoustic model relies strictly on word embeddings to represent words, the model may operate on a different lexicon $\mathcal{D}'$ (say at inference), assuming one can provide corresponding word embeddings $\mathbf{W}'$.

Word embeddings $\mathbf{W}$ are computed with a specific network architecture $f^{wd}(\cdot)$, which operates over sub-word units (in our case characters). The acoustic model $f^{am}(\cdot)$ and word model $f^{wd}(\cdot)$ are trained jointly, forcing both models

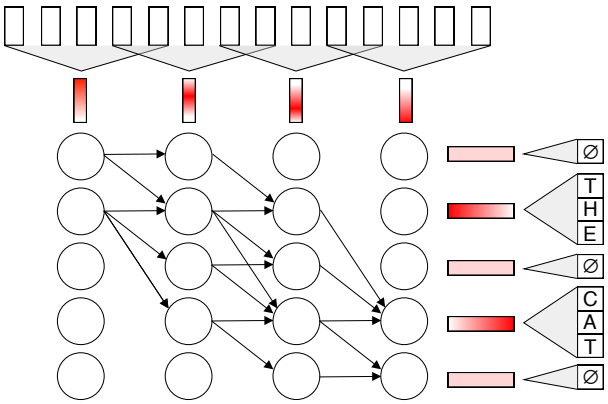


Figure 1: A CTC trained acoustic model combined with the character-based word model. The ∅ denotes BLANK.

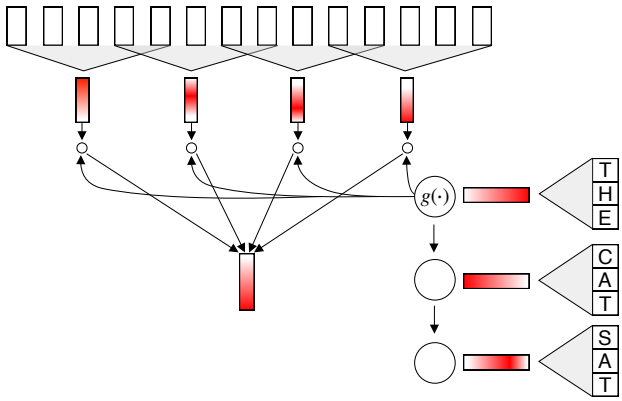


Figure 2: A seq2seq model combined with the character-based word model. The function $g(\cdot)$ denotes the decoder RNN.

to operate in the same acoustically-meaningful d-dimensional embedding space. In the following, we detail the word model, as well as the CTC and seq2seq combined acoustic and word model approaches.

3.1 Character-based Word Modeling

The word model $f^{wd}(\sigma(y))$ maps any word $y \in \mathcal{D}$ into a d-dimensional space. As shown in Figure 3, it takes as input the sequence of characters $\sigma(y)$ of the word, and performs a series of 1D-convolutions interleaved with ReLU nonlinearities and layer normalization [3]. For each input (striding of 1 character), the last convolution outputs a character n-gram embedding (in our case a 7-gram). A fixed-size embedding for each word is obtained by a 1D max-pooling aggregation over these n-gram representations, followed by a linear layer.

One can efficiently compute the embedding of all words in the dictionary by batching several words together. When batching, an extra <PAD> character is used to pad all words to the same maximum word length. Moreover, both CTC and seq2seq-based acoustic modeling rely on special word tokens (like BLANK for CTC or the end-of-sentence token EOS for seq2seq). Embeddings for these special words are also obtained with the same approach, using special <BLANK> and <EOS> tokens as input to the word model.

3.2 Time-Depth Separable Convolutions

Both our CTC and seq2seq architectures use a time-depth separable (TDS) convolution [16]. The TDS block structure decouples integration over time from mixing over channels which allows for much larger receptive fields with a negligible change in the number of parameters. This is particularly important for the direct-to-word approach which needs a larger amount of context for each word prediction.

For a hidden input $\mathbf{H}$ of size $[T, w, c]$, the TDS convolution first performs a $k \times 1$ 2D convolution which results in an output of the same size. The TDS block then resizes this output to $[T, 1, wc]$ and applies two fully connected layers (i.e. 1×1 convolutions) on the hidden dimension wc. The fully connected layers are separated by a ReLU nonlinearity. Residual connections [17] and layer normalization are applied after the convolution and following the fully connected layers. Following the implementation of Hannun et al. [16], our architecture consists of groups of TDS layers separated by convolutional sub-sampling layers.

3.3 CTC-based Acoustic Modeling

Given an input sequence $\mathbf{X}$, the acoustic model is a TDS convolutional network (Section 3.2), which outputs a sequence of embeddings:

$$f_t^{am}(\mathbf{X}) \in \mathbb{R}^d \qquad 1 \leq t \leq T\,. \tag{1}$$

The log-probability $P_t(y|\mathbf{X}, \mathbf{W})$ of a word y at each time step t is then obtained by performing a dot-product with the word embedding of the word model from Section 3.1, followed by a log-softmax operation over possible words in the lexicon $\mathcal{D}$.

$$\log P_t(y|\mathbf{X}, \mathbf{W}) = \mathbf{W}_y \cdot f_t^{am}(\mathbf{X}) - \log \sum_{y' \in \mathcal{D}} e^{\mathbf{W}_{y'} \cdot f_t^{am}(\mathbf{X})}\,. \tag{2}$$

CTC [15] is a structured-learning approach, which learns to align a sequence of labels $\mathbf{Y} = \{y_1, \ldots, y_N\}$ to an input sequence of size T (in our case the embeddings of Equation (1)). An alignment $\pi = \{\pi_1, \ldots, \pi_T\} \in \mathcal{D}^T$ over T frames is valid for the label sequence $\mathbf{Y}$ if it maps to $\mathbf{Y}$ after removing identical consecutive π_t. For example, with $T = 4$, $N = 3$, $\mathcal{D} = \{a, b, c\}$ and a label sequence "cab", valid alignments would be "ccab", "caab" and "cabb". In CTC we also have a BLANK label, which allows the model to output nothing at a given time step, and also provides a way to separate actual label repetitions in $\mathbf{Y}$.

With CTC, we maximize the log-probability $\log P(\mathbf{Y}|\mathbf{X}, \mathbf{W})$, computed by marginalizing over all valid alignments:

$$\log P(\mathbf{Y}|\mathbf{X}, \mathbf{W}) = \log \sum_{\pi \in \mathcal{A}_{\mathbf{Y}}^T} e^{\sum_{t=1}^T \log P_t(\pi_t|\mathbf{X}, \mathbf{W})}, \tag{3}$$

where $\mathcal{A}_{\mathbf{Y}}^T$ is the set of valid alignments of length T for $\mathbf{Y}$. The log-probability of Equation (3) can be efficiently computed with dynamic programming.

It is worth mentioning that the output sequence length T in Equation (1) depends on the size of the input sequence $\mathbf{X}$, as well as the amount of padding and the stride of the TDS architecture. Successful applications of CTC will only be possible if $T \geq N$ (where N is the size of the label sequence $\mathbf{Y}$). When using sub-word units as labels (like characters), it may not be possible to use large strides in the acoustic model. In contrast, since we are using words as labels, N is much smaller than it would be with sub-word units, and we can afford architectures with larger strides – leading to faster training and inference.

3.4 Seq2Seq-based Acoustic Modeling

Given an input $\mathbf{X}$, the seq2seq model outputs an embedding for each possible token in the output:

$$f_n^{am}(\mathbf{X}) \in \mathbb{R}^d \qquad 1 \leq n \leq N\,. \tag{4}$$

The log probability of a word y at step n is obtained with Equation (2). In our case, the embeddings $f_n^{am}(\mathbf{X})$ are computed by first *encoding* the input $\mathbf{X}$ with a TDS convolutional network, and then *decoding* the resultant hidden states with an RNN equipped with an attention mechanism. The encoder is given by:

$$\begin{bmatrix}\mathbf{K} \\ \mathbf{V}\end{bmatrix} = \text{encode}(\mathbf{X}) \tag{5}$$

where $\mathbf{K} = \{\mathbf{K}_1, \ldots, \mathbf{K}_T\}$ are the keys and $\mathbf{V} = \{\mathbf{V}_1 \ldots, \mathbf{V}_T\}$ are the values. The decoder is given by

$$\mathbf{Q}_n = g(\mathbf{W}_{y_{n-1}}, \mathbf{Q}_{n-1}) \tag{6}$$

$$f_n^{am}(\mathbf{X}) = \mathbf{V}_n \cdot \text{softmax}\left(\frac{1}{\sqrt{d}}\mathbf{K}_n^\top \mathbf{Q}_n\right) \tag{7}$$

where $g(\cdot)$ is an RNN. The seq2seq model, like CTC, requires the word embeddings to compute the word-level log probabilities. However, unlike CTC, the word embeddings are also input to the decoder RNN, $g(\cdot)$.

The attention mechanism in the seq2seq model implicitly learns an alignment, thus the final optimization objective is simply the log probability of the word sequence:

$$\log P(\mathbf{Y}|\mathbf{X}, \mathbf{W}) = \sum_{n=1}^{N} \log P(y_n|y_{<n}, \mathbf{X}, \mathbf{W}). \tag{8}$$

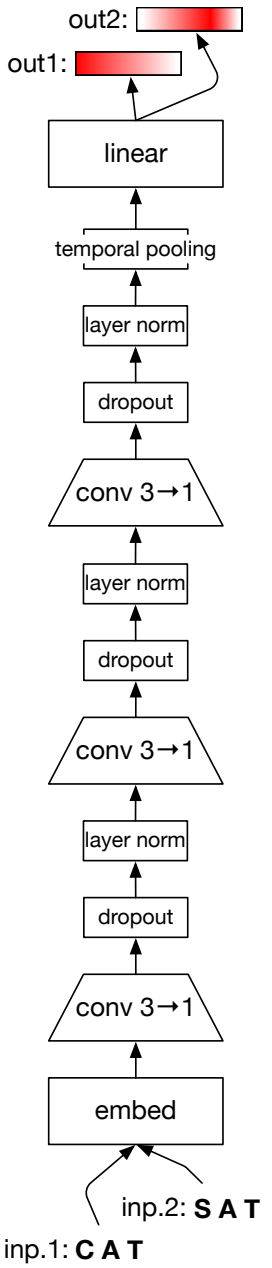


Figure 3: Architecture of the character-based word embedding model.

3.5 Scalable Training via Word Sampling

The resources (computation and memory) required to estimate the posteriors in Equation (2) scale linearly with the number of words in the lexicon $\mathcal{D}$. As is, these word-level approaches simply do not scale to more than a few thousand of words in the lexicon. To circumvent this scaling issue, we propose a simple sampling approach: for each sample $(\mathbf{X}, \mathbf{Y})$ (or batch of samples) we consider a dictionary $\tilde{\mathcal{D}}$ with the size $|\tilde{\mathcal{D}}| \ll |\mathcal{D}|$ fixed beforehand, composed of the labels present in $\mathbf{Y}$, augmented with uniformly sampled labels in $\mathcal{D}$. All operations described in previous sections then remain the same, but performed using $\tilde{\mathcal{D}}$. We will show in Section 4 that this approach is effective in practice.

3.6 Inference and Language Model Decoding

For fast inference, the embeddings $\mathbf{W}$ can be computed in advance only *once*, if the inference dictionary is known beforehand, and fixed. Note that the inference dictionary may or may not be the same as the training dictionary.

As our approach is word-based, there is no need for a decoding procedure at inference time – compared to sub-word unit-based approaches – to obtain a sequence of meaningful words. However, leveraging a word language model trained on a large text corpus can still help to further decrease the word error rate (see Section 4). We implemented a beam-search decoder which optimizes the following objective:

$$\log P(\mathbf{Y}|\mathbf{X}, \mathbf{W}) + \alpha \log P_{LM}(\mathbf{Y}) + \beta|\mathbf{Y}|\,, \tag{9}$$

where $\log P_{LM}(\cdot)$ is the log-likelihood of the language model, α is the weight of the language model, and β is a word insertion weight. Given the acoustic and language models operate at the same granularity (words), the decoder is much simpler to implement than a traditional speech recognition decoder. The decoder is an iterative beam search procedure which tracks hypotheses with the highest probability in the following steps:

- Each hypothesis is augmented with a word in $\mathcal{D}$. For efficiency, only the top K words, according to the acoustic model likelihood, are considered (where K is chosen by the user).
- The individual hypothesis scores are updated according to the acoustic and language model scores, as well as the word insertion score.
- The top B hypotheses are retained (where B is also chosen by the user) and the remaining hypotheses are discarded.

4 Experiments

We perform experiments on the LibriSpeech corpus – 960 hours of speech collected from open domain audio books [26]. Our models are trained on all of the available training data, all hyper-parameters are tuned according to the word error rates (WERs) on the standard validation sets. Final test set performance is reported for both the CLEAN and OTHER settings (the latter being a subset of the data with noisier utterances). We use log-mel filterbanks as features to the acoustic model, with 80 filters of size 25ms, stepped by 10ms. No data augmentation or speaker adaptation was performed. Unless otherwise stated, the lexicon $\mathcal{D}$ at training time contains all the words in the training set (around 89k words) and we sample 5,000 words for each batch, including the labels corresponding to the batch's utterances. Characters used to build the word embeddings include all English letters (a-z) and the apostrophe, augmented by special tokens <PAD>, <BLANK> (for CTC) and <EOS> (for seq2seq), as described in Section 3.1. When decoding with a language model, we use the standard LibriSpeech 4-gram LM, which contains 200k unigrams. When training with a lexicon $\mathcal{D}$ which does not contain all the training words, an additional <UNK> token is used to represent out-of-vocabulary words. We use the open source WAV2LETTER++ toolkit [28] to perform our experiments. We train with Stochastic Gradient Descent, with a mini-batch size of 128 samples split evenly across eight GPUs. Our TDS architectures contain 9 blocks for CTC models and 11 blocks for for seq2seq models with 18m and 37m parameters respectively.

In Table 1, we compare our word-level approach with word piece-based models, which are commonly used with seq2seq style models in speech recognition and machine translation. The word piece models contain 10,000 tokens computed from the *SentencePiece* toolkit [20]. We also show in Table 1 that a model with a stride of 16 gives comparable WERs to a model with a stride of 8, is faster to forward, and is $2\times$ faster to decode.

4.1 Effect of Sampling

In Figure 4, we compare the validation WER on both the CLEAN and OTHER conditions, for a different number of sampled words from the lexicon $\mathcal{D}$, as explained in Section 3.5. We report results for the CTC model. Too few sampled examples makes the training unstable, and slow to converge. We do not see any advantage in having more than 5,000 sampled words per batch. We observe in general that the number of sampled examples may affect the beginning of training – but in our experience, when the sample set is large enough ($\geq$2000), all experiments converge to a similar error rate.

4.2 Importance of Regularization

Our initial runs with the CTC word-level approach were unsuccessful – the loss and WER rapidly diverged after a few iterations. We found that the norm of acoustic and word embedding models output were slowly growing in

Table 1: Comparison of our word-level approaches against a word-piece baseline, with and without language model (LM) decoding. All models have an overall stride of 8 except rows with $s = 16$, which have a stride of 16.

Model	LM	$\|\mathcal{D}\|$ train	$\|\mathcal{D}\|$ test	Dev WER clean	Dev WER other	Test WER clean	Test WER other
word piece seq2seq [16]	None	10k	10k	5.04	14.45	5.36	15.64
word piece CTC	None	10k	10k	6.58	17.52	6.69	18.23
word piece CTC	4-gram	10k	10k	6.05	13.85	6.42	14.81
word-level seq2seq	None	89k	89k	5.99	16.73	6.22	17.32
word-level CTC	None	89k	89k	5.42	15.62	5.63	16.27
word-level CTC, $s=16$	None	89k	89k	5.61	15.49	5.49	16.01
word-level CTC	4-gram	89k	89k	4.35	12.10	4.42	12.82
word-level CTC, $s=16$	4-gram	89k	89k	4.43	12.00	4.51	13.00
word-level CTC	4-gram	89k	200k	**4.09**	**11.12**	**4.26**	**11.98**
word-level CTC, $s=16$	4-gram	89k	200k	4.17	11.21	4.27	12.20

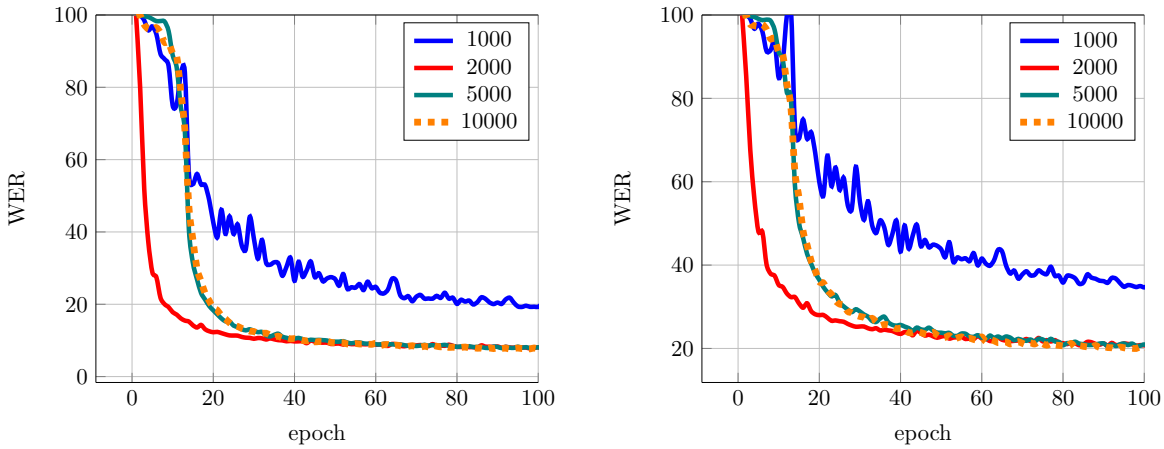


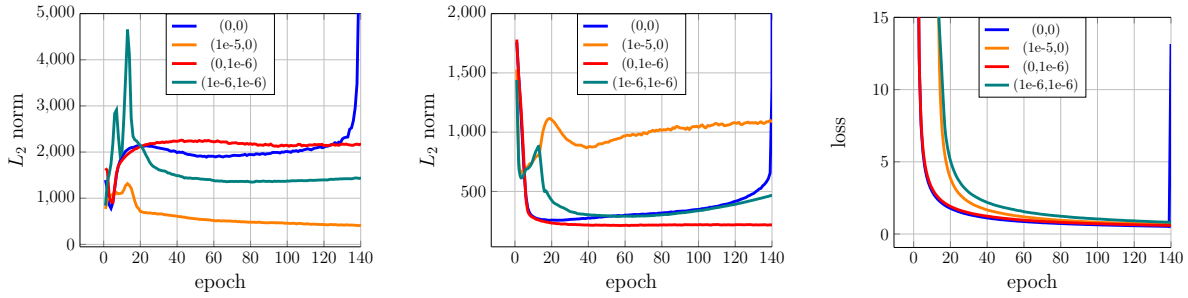
Figure 4: Validation WER with respect to number of training epochs, on the (a) CLEAN and (b) OTHER conditions. Training was with CTC. We show the effect of the number of sampled words (1000, 2000, 5000 and 10000) used in the CTC normalization (see Equation (2)).

magnitude causing numerical instability and ultimately leading to divergence. We circumvent the issue by applying an L_2 regularization on the output of the acoustic or word models. Figure 5 shows that a small weight decay on the acoustic model alone is enough to stabilize the training – all of our subsequent CTC experiments were performed with this regularization.

4.3 Dynamic Lexicon

In Table 1, we show that training with the complete training lexicon (90k words), and then decoding with the language model lexicon shipped with LibriSpeech (200k words) gives a boost in WER. At evaluation time, we simply extended the dictionary, by computing the unseen word embeddings with the character-based word model.

To further evaluate to which extent new words can be added to the lexicon with an evaluation with *no language model*, we trained variants of our model where only the top 10k and 20k words from the lexicon where used at training time. All other words were mapped to the <UNK> word at training. At evaluation, we compared the WER obtained with lexicon the model was trained on, against the WER with the full train (90k) lexicon. On the CLEAN validation set, the WER decreased from 10.70 when restricted to the 10k lexicon that the model was trained on to 8.74 with the full 89k lexicon. Similarly, the WER decreased from 7.10 to 6.35 by increasing the lexicon from the 20k training words to the full 89k words.



(a) (b) (c)

Figure 5: Effect of the L_2 regularization on the output of the acoustic and word models. We show the norm of (a) the acoustic model (AM) output and (b) the word model (WM) output, as well as (c) the training loss, with respect to training epochs. Regularization is denoted as (AM, WM) weights.

4.4 Word Embeddings

We observe the nearest neighbors of common words in the acoustic word embedding space. Acoustically similar words frequently cluster together. We report a few interesting words in Table 2, where we can see that there is a mix of phonetic similarity with semantic proximity. For instance "write"'s neighbours are (in order of proximity) "right" (homophone), "read" (contextually close), "light" and "night" (phonetically close).

Table 2: Nearest neighbors for a few words from the lexicon, in the acoustic word embedding space.

eight:	ate, each, day, night, light
fairy:	very, story, every, fire, fear
write:	right, read, light, night, like
their:	the, they're, your, there, our
each:	its, every, which, any, such
too:	two, so, to, who, true

5 Conclusion

Direct-to-word speech recognition poses an exciting opportunity to simplify ASR models as well as make them more accurate and more efficient. However, this line of research has received little attention due to the difficulty of the problem. Either the model requires massive training sets to learn reasonably sized lexicons or the lexicon is small and unchangeable.

We have demonstrated that a direct-to-word approach for speech recognition is not only possible but promising. Our model gains several advantages from predicting words directly including (1) directly optimizing the word error rate (2) a model which can operate at a much lower frequency and thus is more efficient and (3) a simpler beam search decoder which integrates easily with an externally trained language model. Key to our approach is a character-to-word embedding model which can be jointly trained with an acoustic embedding model. To make this efficient, especially for large vocabulary sizes, we proposed a simple sampling-based mechanism to compute the normalization term needed when training.

We have shown that our approach can be used seamlessly with two commonly used end-to-end models for speech recognition – a CTC trained model and an encoder-decoder model with attention. We validated our models on LibriSpeech, a standard benchmark in read speech recognition. On this data set, the direct-to-word model achieves competitive word error rates. We also demonstrated that our model can accurately predict words never seen in the training set transcriptions.

References

[1] 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 (ICML)*, pages 173–182, 2016.

[2] Kartik Audhkhasi, Bhuvana Ramabhadran, George Saon, Michael Picheny, and David Nahamoo. Direct acoustics-to-word models for english conversational speech recognition. *arXiv preprint arXiv:1703.07754*, 2017.

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*, 2014.

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

[6] Samy Bengio and Georg Heigold. Word embeddings for speech recognition. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[7] Maximilian Bisani and Hermann Ney. Joint-sequence models for grapheme-to-phoneme conversion. *Speech communication*, 50(5):434–451, 2008.

[8] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[9] 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*, pages 4960–4964. IEEE, 2016.

[10] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. 2014.

[11] Ronan Collobert, Christian Puhrsch, and Gabriel Synnaeve. Wav2letter: an end-to-end convnet-based speech recognition system. *arXiv preprint arXiv:1609.03193*, 2016.

[12] Ronan Collobert, Awni Hannun, and Gabriel Synnaeve. A fully differentiable beam search decoder. In *International Conference on Machine Learning (ICML)*, 2019.

[13] Matthew Gibson and Thomas Hain. Hypothesis spaces for minimum bayes risk training in large vocabulary speech recognition. In *Ninth international conference on spoken language processing*, 2006.

[14] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *International Conference on Machine Learning (ICML)*, pages 1764–1772, 2014.

[15] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *International Conference on Machine Learning (ICML)*, pages 369–376, 2006.

[16] Awni Hannun, Ann Lee, Qiantong Xu, and Ronan Collobert. Sequence-to-sequence speech recognition with time-depth separable convolutions. *arXiv preprint arXiv:1904.02619*, 2019.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[18] Stephan Kanthak and Hermann Ney. Context-dependent acoustic modeling using graphemes for large vocabulary speech recognition. In *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages I–845. IEEE, 2002.

[19] Mirjam Killer, Sebastian Stuker, and Tanja Schultz. Grapheme based speech recognition. In *Eighth European Conference on Speech Communication and Technology*, 2003.

[20] Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.

[21] Jinyu Li, Guoli Ye, Amit Das, Rui Zhao, and Yifan Gong. Advancing acoustic-to-word ctc model. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5794–5798. IEEE, 2018.

[22] Tatiana Likhomanenko, Gabriel Synnaeve, and Ronan Collobert. Who needs words? lexicon-free speech recognition. *arXiv preprint arXiv:1904.04479*, 2019.

[23] Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*, 2015.

[24] Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*, 2015.

[25] Andrew Maas, Ziang Xie, Dan Jurafsky, and Andrew Ng. Lexicon-free conversational speech recognition with neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 345–354, 2015.

[26] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

[27] Rohit Prabhavalkar, Tara N Sainath, Yonghui Wu, Patrick Nguyen, Zhifeng Chen, Chung-Cheng Chiu, and Anjuli Kannan. Minimum word error rate training for attention-based sequence-to-sequence models. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4839–4843. IEEE, 2018.

[28] Vineel Pratap, Awni Hannun, Qiantong Xu, Jeff Cai, Jacob Kahn, Gabriel Synnaeve, Vitaliy Liptchinsky, and Ronan Collobert. wav2letter++: The fastest open-source speech recognition system. *arXiv preprint arXiv:1812.07625*, 2018.

[29] Kanishka Rao, Fuchun Peng, Haşim Sak, and Françoise Beaufays. Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4225–4229. IEEE, 2015.

[30] Cicero D Santos and Bianca Zadrozny. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1818–1826, 2014.

[31] Hagen Soltau, Hank Liao, and Hasim Sak. Neural speech recognizer: Acoustic-to-word lstm model for large vocabulary speech recognition. *arXiv preprint arXiv:1610.09975*, 2016.

[32] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems (NIPS)*, pages 3104–3112, 2014.

[33] 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.