RUSSIAN-LANGUAGE SPEECH RECOGNITION SYSTEM BASED ON DEEPSPEECH

O.O. Iakushkin [a], G.A. Fedoseev, A.S. Shaleva, A.B. Degtyarev, O.S. Sedova

Saint Petersburg State University, 7/9 Universitetskaya nab., St. Petersburg, 199034, Russia

E-mail: [a] o.yakushkin@spbu.ru

The paper examines the practical issues in developing a speech-to-text system using deep neural networks. The development of a Russian-language speech recognition system based on DeepSpeech architecture is described. The Mozilla company's open source implementation of DeepSpeech for the English language was used as a starting point. The system was trained in a containerized environment using the Docker technology. It allowed to describe the entire process of component assembly from the source code, including a number of optimization techniques for CPU and GPU. Docker also allows to easily reproduce computation optimization tests on alternative infrastructures. We examined the use of TensorFlow XLA technology that optimizes linear algebra computations in the course of neural network training. The number of nodes in the internal layers of neural network was optimized based on the word error rate (WER) obtained on a test data set, having regard to GPU memory limitations. We studied the use of probabilistic language models with various maximum lengths of word sequences and selected the model that shows the best WER. Our study resulted in a Russian-language acoustic model having been trained based on a data set comprising audio and subtitles from YouTube video clips. The language model was built based on the texts of subtitles and publicly available Russian-language corpus of Wikipedia's popular articles. The resulting system was tested on a data set consisting of audio recordings of Russian literature available on voxforge.com—the best WER demonstrated by the system was 18%.

Keywords: Deep Neural Network, Speech, Neural Network Training

1. Introduction

Currently there are few open source speech recognition systems with close-to-human quality of performance. It is especially true for the Russian language. This paper describes the use of deep neural networks to build an open source speech recognition system. The system is designed to recognize Russian-language speech segments with the average duration of 5-15 seconds in a cleaned dataset. The system is intended to deliver the average WER (Word Error Rate) under 30%.

Mozilla DeepSpeech, an open source end-to-end DNN architecture, was chosen as the starting point for our system. Mozilla DeepSpeech is based on the papers of Baidu Research team [1]. It was launched in May 2016 and reached the lowest WER of 6.5% on the LibriSpeech 'test-clean' dataset for the English language in November 2017. The architecture of Mozilla DeepSpeech is not tied to any particular language and employs the TensorFlow machine learning framework to describe the structure of neural network and optimize computations.

The Word Error Rate (WER) metric is used to evaluate the model's quality at the word level. The WER is a normalized Levenshtein distance between two word sequences that is averaged for all samples. The WER is defined as follows: $WER = (I + D + S) / N$, where I is the number of insertions, D is the number of deletions, S is the number of substitutions, and N is the number of words in the reference.

2. Approach

Many contemporary end-to-end speech recognition systems are based on the Connectionist Temporal Classification (CTC)—an approach that has become a major breakthrough since it was introduced by Alex Graves in [2]. It describes the output layer of the neural network and the loss function computed based on its values. The CTC does not restrict the choice of network architecture and eliminates the need for meticulous data annotation in sequence recognition tasks [3]. In speech recognition, the problem of data annotation lies in the need to align the sequences of audio frames and sequences of letters in conditions where one character may correspond to many audio frames. The CTC method also provides a differentiable function called the CTC loss function: it applies to machine learning by the stochastic gradient descent method.

2.1. Acoustic model

Let us introduce a set of characters A (the alphabet) that has the length of $[A]$ and consists of lowercase letters of the language in question. Let us also consider the alphabet $A' = A\cup\{blank\ character\}$ with the length of $[A'] = n$. The blank character here is required for the CTC method. Let us then consider the input sequence of $\mathbf{x}$ feature vectors and of the length T. Each feature vector in the sequence is the result of mel-frequency cepstral coefficients (MFCC) extraction from a short audio signal that usually has the duration of 20 milliseconds. The extracted MFCC features provide a quantitative description of the audio and will be further used for symbol prediction. The MFCC extraction algorithm allows to define the number of features (let it be m).

Let us also define a recurrent neural network with m inputs, n outputs, and the weight vector ω. The weight vector determines the representation N_ω that transforms the input sequence x to a sequence of length vectors n: $N_\omega : (R^m)^T \rightarrow (R^n)^T$.

2.2. Neural network structure

We use a neural network that consists of five hidden layers. The input layer has m inputs that correspond to m MFCC features derived from a short audio signal. The second and third layers are similar to the first one. As regards the first three layers, the nodes in the neighbouring layers are fully connected and use the activation function called $ReLU$ (Rectifier Linear Unit): $ReLU(x) = max(0,x)$. The forth layer is a bidirectional recurrent layer with LSTM units that uses the hyperbolic tangent as the activation function. The result goes to the fifth layer having $ReLU$ as the activation function. Finally, there is the output layer sized $n = [A']$, where the output value of each node is proportional to the probability of the respective character of the alphabet. Dropout is applied to all five layers with the probability of 0.3 in order to prevent the neural network from overtraining.

3. Training Details

The neural network is trained on aligned audio-transcript pairs. The training dataset is divided into three subsets (train set, dev set and test set) with the proportion of 60:20:20 respectively.

The following algorithm is used for training:

1. The MFCC feature vectors extracted from an audio signal are fed to the input layer of the neural network.

2. The data are transformed by weight values, which results in a prediction matrix of symbols. Each column in the matrix is a probability distribution of characters from the alphabet A' for the time t.

3. The CTC loss function is computed for a particular sample based on the prediction matrix and the expected (reference) transcript. The loss function's value is used to adjust the weights ω in neural network N. The adjustment employs the Adam algorithm, which is a modification of stochastic gradient descent method [4].

4. The described process is reiterated as long as the error value (obtained on a validation set) continues to decrease—that is, the acoustic model is being enhanced in an iterative manner. The neural network's weights are updated based on the CTC loss function's mean value obtained for a group of samples.

The DeepSpeech architecture employs a probabilistic language model to improve the accuracy of speech recognition. This model is essentially a dataset that contains the estimate probabilities of word sequences in a language. Each sequence ranges in length from 1 to N and has a number assigned to it—this number corresponds to the probability of such sequence. Word sequences that consist of n words are called n-grams. The maximum number N that corresponds to the length of a word sequence defines the dimension of the model. If the length of a sequence exceeds the model's dimension or the sequence is not found in a dataset, the probability of such sequence is estimated by reference to the probability of shorter sequences (down to a unigram) it consists of. Mozilla DeepSpeech uses KenLM toolkit [5] to make queries to the language model. The principles of building a probabilistic language model are described in [5; 6].

The language model is used in the beam search algorithm to increase the probability of beams that comprise word sequences used more frequently. It allows to reduce the probability of beams that contain mistakes or unrealistic sequences.

4. Experiments

We used a computer provided by the SPbU Computing Center to make computations required by our task. The computer's technical parameters are as follows:

- CPU: 2 x Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60 GHz;
- RAM: 256 GB;
- GPUs: 2 x Nvidia Tesla P100, 16 GB each.

4.1. Training hyperparameters

The following hyperparameters allow to fine-tune the training process:

1. n_hidden. The number of nodes in the hidden layers of the neural network. It is recommended that this parameter should be defined in accordance with the amount and variety of training data [7]. In Mozilla DeepSpeech, the default value was set at 2048 for the English-language training model. The maximum value is limited by the video memory available.

2. learning_rate. The standard learning rate in Mozilla DeepSpeech is 0.0001.

3. epoch. The number of epochs, where an epoch is one complete forward pass and one complete backward pass of all the training samples in a dataset.

4. train_batch_size, dev_batch_size, test_batch_size – the number of samples in each of the batches used respectively for training, validation and testing. This should be set at the maximum value allowed by the video memory (usually from 8 to 64).

5. dropout_rate – the probability of a node dropout during one training step. This papameter is believed to prevent the network from overtraining [8]. The default value is 0.3.

4.2. Data

Deep neural networks require 500-2000 hours of recorded speech with corresponding transcripts for speech recognition training. We studied several speech sources in Russian and various methods of automatic dataset building. The resulting training dataset includes:

- ‘yt-vad-1k’, a corpus containing about 1000 hours of audio recordings extracted from videos on YouTube that have been created by over 1000 people in a variety of recording conditions and with varying degree of background noise. The corpus also contains both user-provided and auto-generated subtitles (automatic captions) for the extracted audio;
- ‘voxforge-ru-clean’, a corpus containing 11.5 hours of audio from voxforge.org with transcripts;
- ‘yt-vad-650-clean’, a corpus of cleaned audio samples that contains 650 hours of audio.

The audio files are in the ‘wav’ format and have one audio channel (mono), the sample rate of 16,000 Hz, and the depth of 16 bit for each value. The ratio between the audio length and the number of symbols in the transcript is subject to a restriction that follows from the CTC matrix’s decoding algorithm—namely, the number of steps in the CTC matrix must exceed the number of symbols in the transcript

4.3. Training

We obtained the minimum WER of 27% as the result of training on the ‘yt-vad-1k-train’ dataset. The use of the language model allowed to reduce WER by 10% on ‘voxforge-ru-clean-test’ dataset.

The model trained on the ‘yt-vad-650-clean’ corpus—which was cleaner than ‘yt-vad-1k’—showed a slight increase in the minimum WER. It can be explained by a smaller amount of data—650 hours compared to 1000 hours in the first experiment. However, testing with the language model delivers better WER in both experiments.

We managed to reduce the minimum WER from 27% down to 22% by means of further training the model already pre-trained on ‘yt-vad-1k’.

Table 1. Training Evaluation

		WER, %			
		Testing without LM		Testing with LM	
		yt-vad-650-clean-test	voxforge-ru-clean-test	yt-vad-650-clean-test	voxforge-ru-clean-test
Train dataset	yt-vad-1k-train	35.17	37	35.2	27
	yt-vad-650-clean-train	37.5	39.8	35	28
	yt-vad-1k-train + yt-vad-650-clean-train	35.3	33.2	33	22

5. Acknowledgement

This research was supported by SPbU (Saint Petersburg State University) grant no. AAAA-A18-118071790047-9 (id: 28612502). The authors would like to acknowledge the Reviewers for the valuable recommendations that helped in the improvement of this paper.

6. Conclusion

We have developed a Russian-language speech recognition system that delivers the WER of 22% on the 'voxforge-ru-clean-test' dataset. We have employed it to perform speech-based text search in a large collection of video. This system can be useful in a variety of tasks involving speech-based search, though a higher recognition accuracy is required in such tasks as speech recognition by voice assistants. The improvement of speech recognition accuracy will be the goal of further research.

References

[1] Hannun A., et. al Deep speech: Scaling up end-to-end speech recognition // arXiv preprint arXiv:1412.5567. — 2014.

[2] Graves A., et. al Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks // Proceedings of the 23rd international conference on Machine learning. ACM. 2006, pp. 369-376.

[3] Graves A. Supervised sequence labelling // Supervised sequence labelling with recurrent neural networks. Springer, 2012, pp. 52—73.

[4] Kingma D. P., Ba J. Adam: A method for stochastic optimization // arXiv preprint arXiv:1412.6980. — 2014.

[5] Heafield K. KenLM: Faster and smaller language model queries // Proceedings of the Sixth Workshop on Statistical Machine Translation. — Association for Computational Linguistics. 2011., pp. 187-197.

[6] Heafield K., et al. Scalable modified Kneser-Ney language model estimation // Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Vol. 2, 2013., pp. 690-696

[7] Weigend A. On overfitting and the effective number of hidden units // Proceedings of the 1993 connectionist models summer school, Vol. 1, 1994, pp. 335-342.

[8] Srivastava N., et al. Dropout: A simple way to prevent neural networks from overfitting // The Journal of Machine Learning Research 2014, Vol. 15, pp. 1929-1958.