BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

http://fit.vut.cz

Project practice
2021/2022

Offline ASR system for voice assistant based on RNNT architecture

Skuratovich Aliaksandr

Abstract
The problem of low-latency mobile speech recognition systems for voice assistants was studied. Also, three three models with the RNNT architecture were trained, two of which showed good results in low-resource languages such as the Czech language. The comparison of the models and its sizes was performed.
Dataset used in the model training is Common Voice 7.0 Czech language corpus. The characters distribution in the dataset and in the Czech language was compared to find out the problem that originally caused high WER (over a 100%).

Keywords: end-to-end — RNNT — ASR

Supplementary Material: *N/A*

xskura01@vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

Due to the wide use of smart devices and the popularity of AI, smart assistants became an essential part of everyday life. However, there is a big challenge to port a state-of-the-art speech recognition system onto computationally constrained devices such as smartphones.

Many speech recognition software used by the voice assistances works as follows: The voice is transferred to the server, where the computations are performed and target speech is recognized by computational intensive neural networks. The main disadvantage of this approach is that there is a need to have an internet connection even for a simple command such as opening an application or starting playing the music.

Therefore, the main aim of mobile speech recognition and voice assistants is to simplify the ASR system as possible without losing much accuracy to be able to perform these tasks fast and without an internet connection. For these purposes, the state-of-the-art neural network architecture can be used. It's called RNNT or Recurrent Neural Network Transducer and it is able to recognize your speech offline character by character, as people do.

In this article, three different ASR systems of different sizes implemented using RNNT architecture will be compared. The problem of Automatic Speech Recognition for mobile devices will be introduced.

The task was to study and train the RNNT model to then implement an ASR model suitable for the mobile voice assistant.

1.1 RNN

To introduce the RNNT model, we first need to know what the RNN part is. RNN stands for Recurrent Neural Network, you can see its unfolding through time on the Figure 1. Basically, RNN can be imagined as an IIR filter[1]: the output $\mathbf{z}_t$ of time t is computed based on the input of the time t $\mathbf{x}_t$ and output of the time $t-1$. RNNs are good for working with sequential data, where the inputs are connected through time, so there is a dependence between inputs, that leads to the dependence between outputs. Therefore, to store the information about the previous inputs there is a need to have some kind of memory.

[1] https://en.wikipedia.org/wiki/Infinite_impulse_response

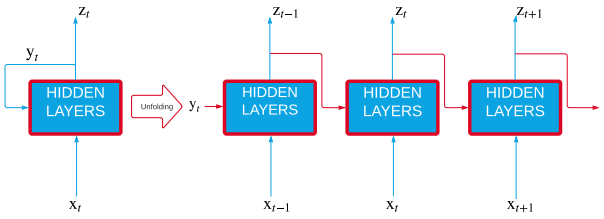

Figure 1. RNN: Recurrent Neural Network

1.2 RNN-Transducer

RNNT or Recurrent Neural Network Transducer architecture was introduced by Alex Graves in 2012 in the paper "Sequence Transduction with Recurrent Neural Networks" [3]. Then in 2019 in the paper "Streaming End-to-end Speech Recognition For Mobile Devices" Google Research team showed that the model can be improved and optimized to fit mobile devices without losing accuracy [4].

The Figure 2 represents the diagram of the RNNT model, which consists of Prediction, Encoding and Join networks.

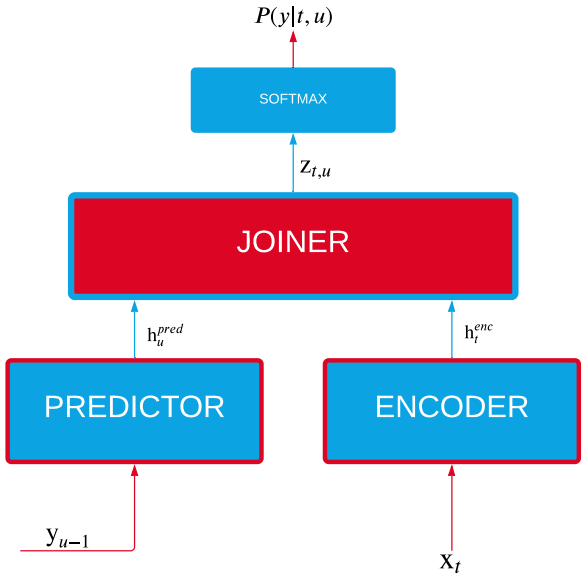

Figure 2. RNN-Transducer architecture

1. **Encoder** network is a bidirectional LSTM [8] network converting a vector of input features to the encoded representation. It works similar to the AM (Acoustic Model)[2].
2. **Predictor** or a prediction network is a causal network (such as RNN) that takes as an input the previous outputs and produces features that can be used to predict the next output. The prediction network works similar to the LM (Language model)[3]
3. **Joiner** is a feed-forward network that combines the encoded vector h_t^{enc} and the predicted vector h_u^{pred}, and outputs them into a softmax $z_{t,u}$ to compute the output probabilities over all output symbols with a special <unk> symbol means Null.

The principle of operation of a RNNT is similar to a recurrent neural network, because RNN-Transducer returns probabilities depending on the input and its previous output.

2. Dataset

Common Voice is a project from Mozilla, that provides free datasets for more than 40 languages, including the Czech language. The Common Voice corpus was introduced in the paper named “Common Voice: A Massively-Multilingual Speech Corpus” [2]. Common Voice 7.0, contains 54 hours of Czech spoken language, which is not much, however, it is enough to see the RNNT performance in practice.

In 2019, it contained 39 languages and 2500 hours of collected audio. Now, there are more 13,000 hours of the human voice and 76 languages in the Common Voice corpus.

The Common Voice platform itself provides an environment to participate in creating and improving its datasets, which significantly helps in training ASR systems on the low-resource languages.

Also, one of the speechbrain recipe for RNNT uses the Common Voice dataset to train the transducer model. However, the recipe was not implemented for the Czech language, so there was a need to change the preparation step of the dataset to make it possible to work with the Czech language.

set	**train**	**dev**	**test**
duration [hours]	14.14	8.21	8.09

Table 1. Common Voice 7.0 for the Czech language

As you can see, the dataset itself is a small one, but even though the RNNT network achieved good results on it. It is worth mentioning that RNNT architecture is suitable for training on small datasets (in the original paper, the Transducer was trained on TIMIT speech corpus, which has about 5.4 hours of speech) [3].

Character distribution in the CV dataset (see Figure 3) and in the Czech language look pretty much the same, even if the dataset itself is not very big, so the possibility of some characters to prevail is higher.

It is worth mentioning, that due to the ’ch’ grapheme was not encountered when I was analyzing the dataset, I decided to distribute occurrences of ’ch’ grapheme

[2] https://en.wikipedia.org/wiki/Acoustic_model
[3] https://en.wikipedia.org/wiki/Language_model

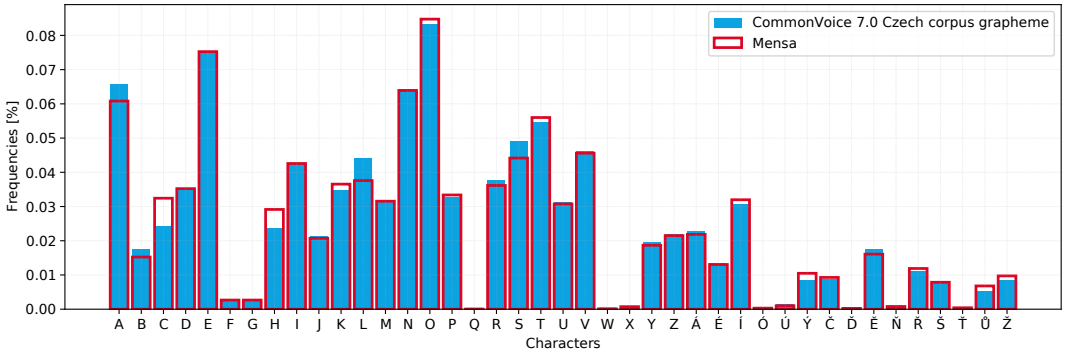

Figure 3. Comparison of Mensa distribution of characters in Czech language and distribution from the Common Voice 7.0 dataset. Both are normalized.

between 'c' and 'h' while comparing the distribution of characters in the Czech language and in the dataset. Because some graphemes are represented in the dataset very rarely, I replaced some of them with graphemes that sound similar. Also, there were non-Czech letters (for example, French, German names, or cities). These letters have also been replaced by the Czech ones.

3. Training

All models were implemented using the speechbrain toolkit, which was introduced in the "SpeechBrain: A General-Purpose Speech Toolkit" paper [7]. In this toolkit, many recipes are implemented, including RNNT. Also, the presence of recipes makes it possible to explore different neural architectures, and the data processing pipeline prepared by them (called the "datio pipeline") helps not to waste time on the preparation of datasets.

Several models were trained on the Czech Common Voice dataset. Then, some hyperparameters were changed to perform better on the given language.

A model provided by a Speechbrain with a default token type (first, data are tokenized using SentencePiece tokenizer [6] with unigram language model [5]) had been causing imprecise results (s.t. the WER[4] and CER were above 100%).

Because the Common Voice Czech dataset is very small for such a complicated language as Czech, unigram encoding was not a good choice. Therefore, there were attempts to use byte pair and character tokens instead. A few models were trained using bpe encoding, but none of them showed better results than the 'character' ones. As for the character encoding, it was the most successful choice both at WER and CER and it showed a significant decrease in both metrics.

[4]https://towardsdatascience.com/evaluating-ocr-output-quality-with-character-error-rate-cer-and-word-error-rate-wer-853175297510

Then, I decided to go forward and reduce the model size to see how it affects the model performance. Therefore I changed some of the parameters of the RNNT parts (encoder, predictor, and joiner) such as number of layers, number of neurons, and optimizers.

Networks x metrics	**#params [M]**	**WER**	**CER**
RNNT-baseline	138.4	55.8	19.8
RNNT-baseline-s	6.1	64.53	22.57
RNNT-baseline-xs	1.7	74.54	28.12

Table 2. Comparison of three trained networks

Here, **RNNT-baseline** model has all the parameters from the speechbrain RNNT recipe. **RNNT-baseline-xs** is a model where each component has at least twice smaller than the default one, and **RNNT-baseline-s** has the same encoder implemented as bi-directional CRDNN with two LSTMs, but with a greater number of CNN channels and hence the larger CNN matrix.

In the Figure 4, we can see the training statistics of three different implementations of RNN-Transducers.

As it can be expected, the best implementation is the **RNNT-baseline** with 138.4 million parameters. However, at the beginning of the training, you can see the **RNNT-baseline-s** model shows better CER than the default implementation. When it comes to WER though, the default model outperforms **RNNT-baseline-s**. I assume, that it is mainly caused by the size of the joiner network, which was simply not able to recognize where the word ends. The **RNNT-baseline-xs** model didn't perform well, but the fact that CER was increased by 10%. The fact that the number of parameters was reduced by almost 80 times shows that future improvements in size are possible and will lead to better results.

Surprisingly, model **RNNT-baseline-s** showed the best results in terms of training loss and validation loss, while the **RNNT-baseline** had a noticeable overfitting started the 15th epoch.

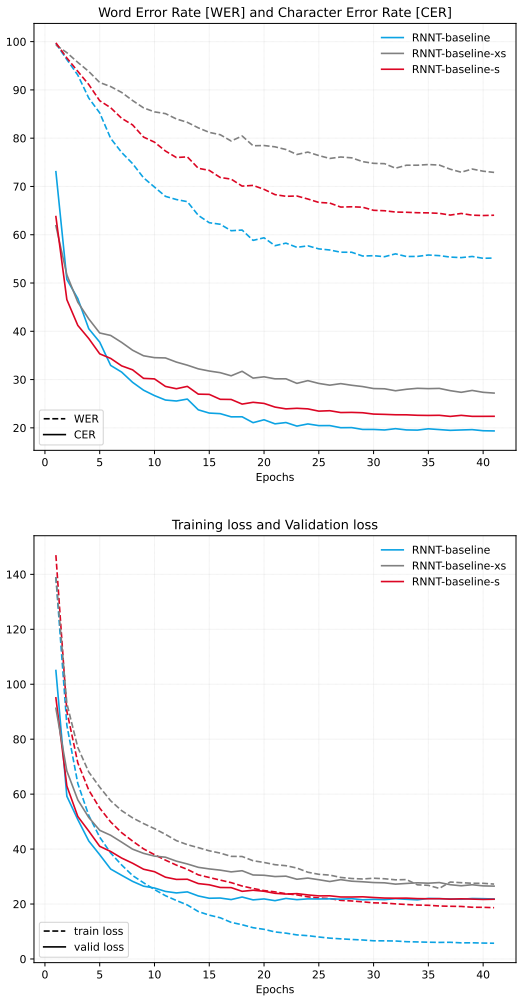

Figure 4. Learning curves for various models

My guess is that this is because the large model was just able to memorize this small training set, while the medium and small models learned to recognize speech because they don't have enough memory.

4. Conclusions

4.1 Results

Model training on a small dataset for a complex language showed good results, which can be improved by integrating the language model. The architecture also showed that reducing its size by 22 times increased the WER by 10%, but the CER was increased only by 2.77%, which may mean that the model correctly recognizes single phonemes, but cannot compose words from them, which may be due to the small join with a layer or a small predictor, while the encoder works in much the same way and reducing its size does not greatly affect the result

Also, I have to mention, that I have learned about ASR in general, trained several systems, and perform an analysis on how they work dependent on configurations. Also, I have understood the state-of-the-art RNNT architecture, so I am sure, that my future development in speech recognition will be faster and more successful.

4.2 Future work

Firstly, since the Common Voice 8.0 dataset has been released, I plan to experiment with more ASR system within this dataset and compare them with the results that were obtained during training on the previous version.

Further goals are to continue to study the theory around automatic speech recognition and around RNNT models to gain more knowledge for future work. Also, there are various papers that show that there is a possibility to improve the model. For example, using FastEmit [9] or parameter quantization presented in [1].

However, I'm sure that a better study of both these problems and neural architectures in general will help to achieve significant improvements. After being implemented and optimized, the model will be used as an ASR system in the mobile voice assistant.

As an opportunity for further work, it may be possible to use this architecture to create a Czech voice assistant, where there will be no such restrictions on the size of the model because it will run on the server.

Acknowledgements

I would like to thank Martin Karafiát for providing the opportunities to participate in this project, Petr Schwarz for organizing meetings and explaining the theory behind ASR systems and Jan Švec for helping me to understand a lot of things.

References

[1] ALVAREZ, R., PRABHAVALKAR, R. and BAKHTIN, A. *On the efficient representation and execution of deep acoustic models*. 2016.

[2] ARDILA, R., BRANSON, M., DAVIS, K., HENRETTY, M., KOHLER, M. et al. *Common Voice: A Massively-Multilingual Speech Corpus*. 2020.

[3] GRAVES, A. Sequence transduction with recurrent neural networks. *ArXiv preprint arXiv:1211.3711*. 2012.

[4] HE, Y., SAINATH, T. N., PRABHAVALKAR, R., MCGRAW, I., ALVAREZ, R. et al. *Streaming End-to-end Speech Recognition For Mobile Devices*. 2018.

[5] KUDO, T. *Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates*. 2018.

[6] KUDO, T. and RICHARDSON, J. *SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing*. 2018.

[7] RAVANELLI, M., PARCOLLET, T., PLANTINGA, P., ROUHE, A., CORNELL, S. et al. *SpeechBrain: A General-Purpose Speech Toolkit*. 2021.

[8] STAUDEMEYER, R. C. and MORRIS, E. R. Understanding LSTM - a tutorial into Long Short-Term Memory Recurrent Neural Networks. *CoRR*. 2019, abs/1909.09586. Available at: http://arxiv.org/abs/1909.09586.

[9] YU, J., CHIU, C.-C., LI, B., CHANG, S. yiin, SAINATH, T. N. et al. *FastEmit: Low-latency Streaming ASR with Sequence-level Emission Regularization*. 2021.