NLP Course Project
End-to-End ASR system for recognition Russian speech

Andrey Gorbachev

May 2022

Abstract

The goal of the project is to implement ASR, based on the End-to-End Architecture, for Russian speech recognition. Automatic speech recognition, or ASR, is the use of machine learning or artificial intelligence (AI) technology to process human speech into readable text. In contrast with conventional hybrid ASR systems, which consist in multiple complex modules such as acoustic, lexicon, and language models, End2End systems typically use a single deep network trained to directly map an input audio sample to a sequence of words or characters, alleviating the need for expert knowledge to build competitive systems The document describes the enlarged structures of the ASRS used, additions and changes to the baseline and results of research. The document also contains a brief description of the Russian datasets used. The link to project code is here: `https://github.com/Andrey-Gorbachev/ODS-NLP-course-Project`.

1 Introduction

ASR systems are used in popular applications such as chatbots, messengers, voice assistants. They are also useful for transcribing audio recordings and direct colloquial speech.

Two systems were considered in the project: Seecharran toolset and Keras Speech Recognition Transformer.

SpeechBrain is good because, in addition to SotA results in ASR, it is a comprehensive multitasking tool that can be integrated into Pipeline to implement various voice processing tasks with state-of-the-art (SotA) baselines

In addition, SpeechBrain is an end-to-end ASR that does not require a preliminary acoustic model and alignments for its implementation. What distinguishes it from, for example, Kaldi and CMUSphinx.

An important goal of the project is to try Russian datasets in pipeline SpeechBrain, while no such solutions have been found on the Internet.

The Karas-based system is simpler and gives results faster on a simpler data set.

2 Related Work

SpeechBrain ASR for LibriSpeech test-clean dataset shows Word Error Rate (WER%):
- 2.91 in CTC+Att CRDNN GRU architecture (Technique Encoder Decoder)
- 2.46 in CTC+Att Transformer GRU architecture.
[8] SpeechBrain: A General-Purpose Speech Toolkit.

This is at the level of modern SotA speech recognition toolkits with WER from 1.4 to 8.0, Speech Recognition on LibriSpeech test-clean [9], "WER are we?"[11]

[8]:"A few other toolkits support multiple speech tasks. Of these, the ones we (SpeechBrain) consider most related to SpeechBrain are Fairseq [4], NeMo [5], and ESPnet [3]. Fairseq is developed by Facebook to support sequence-to-sequence processing. It includes models such as ConvS2S [2], transformers [1], and wav2vec [7]. However, speech processing encompasses several paradigms outside of sequence-to-sequence modeling. SpeechBrain also supports regression tasks (e.g., `speech enhancement, separation`), classification tasks (e.g., `speaker recognition), clustering (e.g., diarization)`, and even signal processing techniques (e.g., `multi-microphone combination)`.

NeMo is a toolkit for conversational AI developed by NVIDIA, which provides useful neural modules for many speech processing tasks, including speech recognition, speaker diarization, voice-activity detection and text-to-speech. Due to its industrial orientation, NeMo offers efficient ready-to-use models, such as Jasper [63], QuartzNet [**64**], and Citrinet [10].

SpeechBrain also provides several ready-to-use models, but focuses more heavily on research and education by providing a wide variety of baselines, models, and recipes that users can easily inspect and modify in the experiments.

ESPnet, in its current form, is the closest toolkit to SpeechBrain. Both are academically driven and support numerous speech tasks. ESPnet started as an end-to-end speech recognition library and progressively grew to support different tasks.

By contrast, SpeechBrain to address a wide variety of tasks from the outset. This means that combining technologies and developing recipes for new tasks is extremely simple."
Karas based system shows comparable results on a data set with one speaker. At the same time, it has a simple architecture and implementation, learns faster.

3 Model Description

3.1 SpeechBrain

SpeechBrain ASR system is composed of 3 different but linked blocks:

- Tokenizer (unigram) that transforms words into subword units and trained with the train transcriptions of LibriSpeech.
- Neural language model (RNN LM or Transformer LM) trained on the full

380K words russian dataset.
- Acoustic model - in our project:
- variant 1. Acoustic model (CRDNN[1] + CTC/Attention). The CRDNN architecture is made of N blocks of convolutional neural networks with normalisation and pooling on the frequency domain. Then, a bidirectional LSTM is connected to a final DNN to obtain the final acoustic representation that is given to the CTC and attention decoders.
- variant 2. Acoustic model made of a transformer encoder and a joint decoder with CTC + transformer. Hence, the decoding also incorporates the CTC probabilities.

The system is trained with recordings sampled at 16kHz (single channel). The code will automatically normalize your audio (i.e., resampling + mono channel selection) when calling transcribe-file if needed.

In the figure Fig. 1 is an example of a typical speech recognition pipeline used in SpeechBrain:

The speech recognition process starts from the `raw waveform directly`.

The original waveform is contaminated with different `speech augmentation techniques` such as time/frequency dropout, speed change, adding noise, reverberation, etc. These disturbances are activated randomly according to some probabilities specified by the user. They are also added on-the-fly without the need of storing the augmented signals on disk.

ASR then extract speech features, such as Short-Term Fourier Transform (STFT), spectrograms, FBANKs, and MFCCs. Also, the features can be computed on the fly thanks to a very efficient GPU-friendly implementation.

ASR then feed the features into the speech recognizer, which is a neural network that maps the input sequence of features into an output sequence of tokens (e.g., phonemes, characters, subwords, words). SpeechBrain supports popular techniques such as Connectionist Temporal Classification (CTC), Transducers, or Encoder/Decoder with attention (using both RNN- and Transformer-based systems).

The posterior probabilities over the output tokens are processed by a beam-searcher that explores different alternatives and outputs the best one. The alternatives can be optionally rescored with an external language model that could be based on RNN or transformers.

Now is more in detail the different technologies supported for speech recognition:

3.1.1 Connectionist Temporal Classification (CTC)

CTC is the simplest speech recognition system available in SpeechBrain.

For each time step, it outputs a prediction. CTC adds a special token called blank. That allows the network to output nothing when not sure about what to emit. The CTC cost function uses dynamic programming to align over all the possible alignments.

[1] C - CNN, R - RNN, D - Dense, NN - Neuron Net

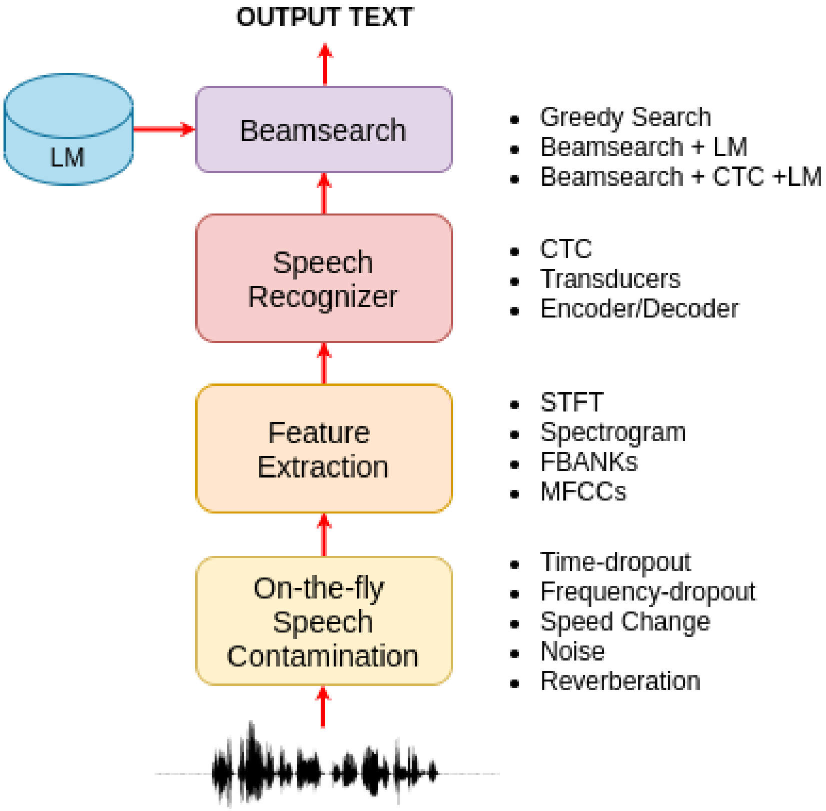


Figure 1: ASR overview

For each alignment, we can compute a corresponding probability. The final CTC cost is the sum of the probabilities of all possible alignments. This can be computed efficiently using the forward algorithm (the one described in the Hidden Markov Model literature, not the one used for neural networks).

In the encoder-decoder architectures, we use attention to learn the alignment between input-output sequences. In CTC, we do not learn it, but we integrate over all the possible alignments.

Essentially, CTC can be implemented through a special cost function plugged on top of the speech recognizer (often but not always based on recurrent neural networks).

3.1.2 Transducers

Transducers As shown in the Figure 2, Transducers augment CTC by adding an autoregressive predictor and a join network. An encoder (usually based on

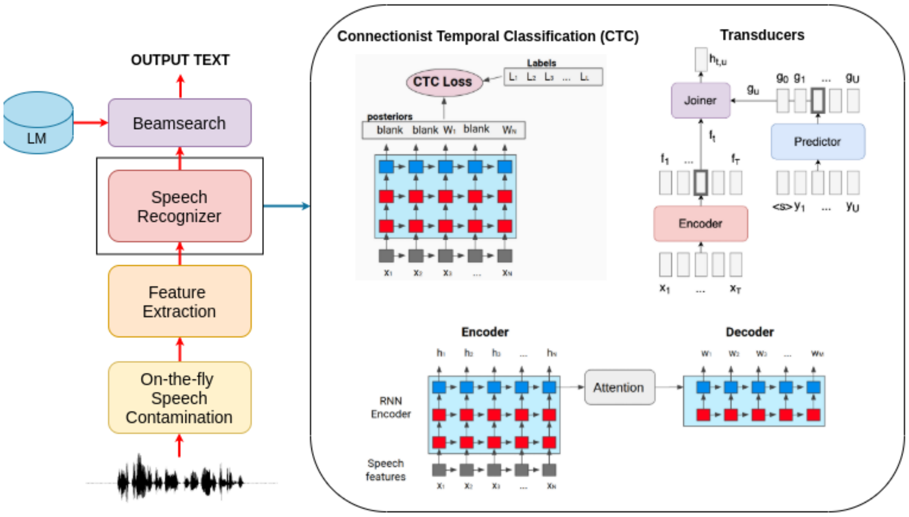


Figure 2: ASR Technologies

RNNs) transforms the input features into a sequence of encoded representations. The predictor, instead, generates a latent representation based on the previously emitted outputs. A join network combines the two and a softmax classifier predicts the current output token. During training, CTC loss is used after the classifier.

Transduser is better version for ASR [**CTCvsTransduser**], bit i didn't have time to experiment with this model in the project, however, this is an interesting groundwork for the future

3.2 Keras Transformer ASR

Karas architecture, unlike the previous system:

- Uses a simple tokenizer/vectorizer - without using and training a neural network to select pieces of words. Vocabulary consists of letters and special characters.
- Does not have a Language Model.
- Uses Transformer with traditional encoder and decoder with attention - for Acoustic Model.

4 Dataset

4.1 Russian Dataset for SpeechBrain

Russian Speech Datasets are provided by Microsoft Corporation with CC BY-NC license.
The CC BY-NC license requires that the original copyright owner be listed as the author and the work be used only for non-commercial purposes

A collection of speech samples from various audio sources. The dataset contains short audio clips in Russian.

That set of Russian Speech to Text (STT) data includes:

16 million statements 20,000 hours 2.3 TB (without compression in the format .wav in int16), 356 GB in opus All files are converted to opus, with the exception of test datasets . The main purpose of the dataset is to train speech-to-text conversion models.

For instructions on downloading the dataset directly, see the GitHub download instructions page `https://github.com/snakers4/open_stt`.

buriy-audiobooks-2-val was chosen as the main dataset, which contains 7,850 rtterances, 4.9 Hours of total recording time, 1 Gb of volume and 99% accuracy with manual annotation.

To embed this dataset into the SpeechBrain pipeline, we had to prepare the prepare_ru_dataset module

4.2 Russian Dataset for Keras ASR

For Keras ASR uses Russian LibriSpeech (RuLS) `https://www.openslr.org/96/` The dataset was created by NVIDIA CORPORATION, 2020. The dataset contains audio with:

- BOOK_ID: Book ID in the LibriVox project ID
- CHAPTER_ID: The ID of the chapter on LibriVox
- CHAPTER_TITLE: The title of the chapter on LibriVox, 'ALL' - means the whole book is read by the same reader
- READER_ID: The ID of the reader in the LibriVox's database
- READER_NAME: The NAME of the reader in the LibriVox's database
- GENDER: Reader's gender
- SUBSET: The corpus subset to which this chapter is assigned

Selected for the experiment subdataset:

- 2.4G - comparable in size to the data set from baseline Keras
- 8169 - Reader_name tovarisch
- 13240/ALL - Book_ID/CHAPTER_TITLE Book Обрыв (Obryv) /ALL

5 Experiments

This section should include several subsections.

5.1 Metrics

Standard metrics were used in the experiments: WAR and CER.

`Word error rate (WER)` `https://en.wikipedia.org/wiki/Word_error_rate` is a common metric of the performance of a speech recognition or machine translation system. The general difficulty of measuring performance lies in the fact that the recognized word sequence can have a different length from the reference word sequence (supposedly the correct one).

The WER is derived from the Levenshtein distance, working at the word level instead of the phoneme level. The WER is a valuable tool for comparing different systems as well as for evaluating improvements within one system. This kind of measurement, however, provides no details on the nature of translation errors and further work is therefore required to identify the main source(s) of error and to focus any research effort. This problem is solved by first aligning the recognized word sequence with the reference (spoken) word sequence using dynamic string alignment. Examination of this issue is seen through a theory called the power law that states the correlation between perplexity and word error rate.

Word error rate can then be computed as:
$WER = \frac{S+D+I}{N} = \frac{S+D+I}{S+D+C}$
where
• S is the number of substitutions,
• D is the number of deletions,
• I is the number of insertions,
• C is the number of correct words,
• N is the number of words in the reference (N=S+D+C)

The intuition behind 'deletion' and 'insertion' is how to get from the reference to the hypothesis. So if we have the reference "This is wikipedia" and hypothesis "This _ wikipedia", we call it a deletion.

`Character error rate (CER)` **??** is a common metric of the performance of an automatic speech recognition system. CER is similar to Word Error Rate (WER), but operates on character instead of word.

Character error rate can be computed similar WER - as:
CER = (S + D + I) / N = (S + D + I) / (S + D + C)
CER's output is not always a number between 0 and 1, in particular when there is a high number of insertions. This value is often associated to the percentage of characters that were incorrectly predicted. The lower the value, the better the performance of the ASR system with a CER of 0 being a perfect score.

5.2 Experiment Setup

General design of experiments:

1. Baseline, like basic pipeline, with English dataset (Minilibrispeech) and pretrained model. Base architecture: RNN Language Model & CRDNN / CTC /Attention Acoustic Model.

2. Baseline with base architecture and Russian dataset (buriy-audiobooks-2-val) and 'zero' model. Base architecture: RNN Language Model & CRDNN/ CTC /Attention Acoustic Model.

Analysis of Seecharran experiments showed that the best learning rate when using the Transformer architecture for the Language Model and for the Acoustic Model.
Therefore, next experiment:

3. Modified baseline: Transforner Language Model & Transformer Acoustic model (transformer encoder and a joint decoder with CTC + transformer) & Tiny Russian dataset (buriy-audiobooks-2-val) - one item from dataset for train, valid, test data.

4. Modified baseline: Transforner Language Model & Transformer Acoustic model (transformer encoder and a joint decoder with CTC + transformer) & Russian dataset (buriy-audiobooks-2-val).

5. Ehere was an experiment and by mistake: pre trained english RNNLM & CRDNN/CTC/Attention or Inference russian audio.

Additionally, the pipeline was adjusted to save and use the generated models on HuggingFace.

6. Keras baseline, adapted for the use of the Russian dataset and supplemented with prediction functionality with audio output and evaluation functionality.

5.3 Baselines: Base architecture RNNLM/CRDNN with english dataset miniLibriSpeech, pretrained models

Baseline pipline:
Architecture used:
- Tokenizer (unigram) that transforms words into subword units and trained with the train transcriptions of LibriSpeech. SpeechBrain relies on the popular SentencePiece for tokenization
- Neural language model (RNNLM) trained on the full 10M words dataset.
- Acoustic model (CRDNN + CTC/Attention). The CRDNN architecture is

made of N blocks of convolutional neural networks with normalisation and pooling on the frequency domain. Then, a bidirectional LSTM is connected to a final DNN to obtain the final acoustic representation that is given to the CTC and attention decoders.

5.4 Baseline with russian dataset buriy-audiobooks-2-val

The same pipeline, but without the pretrained model and with our own recipe for downloading and preprocessing russian audio and transcription data.

5.5 Transforner Language Model & Transformer Acoustic model & Russian dataset buriy-audiobooks-2-val

The same pipeline, but without the pretrained model and with our own recipe for downloading and preprocessing russian audio and transcription data. Language and Acoustic Models replaced by Transformer variants.

5.6 Transforner Language Model & Transformer Acoustic model & tiny Russian dataset buriy-audiobooks-2-val

Ehe same pipeline as the previous one.
However, only one record was used from the dataset. The training was conducted on the CPU

5.7 Baseline with inference by russian dataset buriy-audiobooks-2-val

Similar to the baseline, with the pretrained model, with our own recipe for downloading and preprocessing Russian audio and transcription data. Inference russian audio.

5.8 Keras baseline, adapted for the use of the Russian dataset

Compared to previous experiments, here, in addition to architecture, a dataset with a single speaker was used. This allowed us to speed up calculations and get more meaningful results.
The functionality for downloading and preprocessing the Russian dataset was developed..
Additionally, the pipeline was adjusted to save and use the generated models on HuggingFace.

6 Results

6.1 Baselines: Base architecture RNNLM/CRDNN with english dataset miniLibriSpeech, pretrained models

When training the network, the first epoch gave the result:

- valid loss: 1.39, valid CER: 2.21e+02, valid WER: 2.23e+02.

For 4 epochs of training (each for about 40 minutes on a Tesla 1xK80x24Gb processor, the indicators reached the following values:

- valid loss: 1.37, valid CER: 2.29e+02, valid WER: 2.30e+02

This is the result that alerted.

These indicators are important for comparison with the launch of the Russian dataset.

6.2 Baseline with russian dataset buriy-audiobooks-2-val

6.2.1 Acoustic Model

Due to limitations on the power used by video cards, it was not possible to achieve significant results. However, they correspond to the dynamics of the basic version. It was possible for 12 epochs of 35-40 minutes each) to achieve a decrease in WER / SER by 2 times. Fig. 3

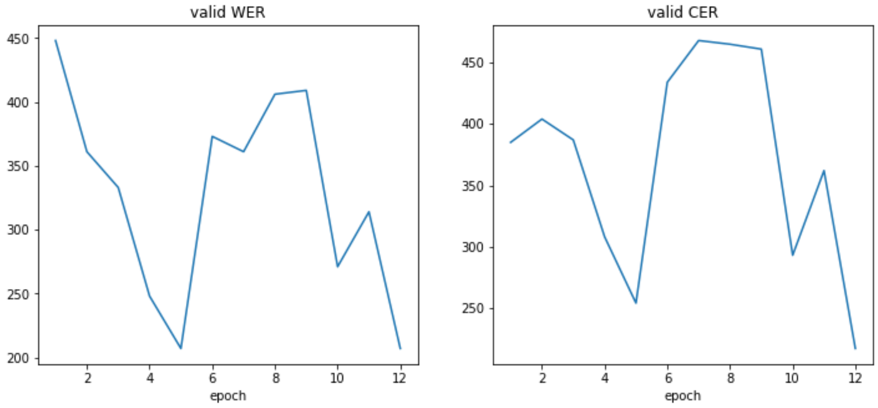
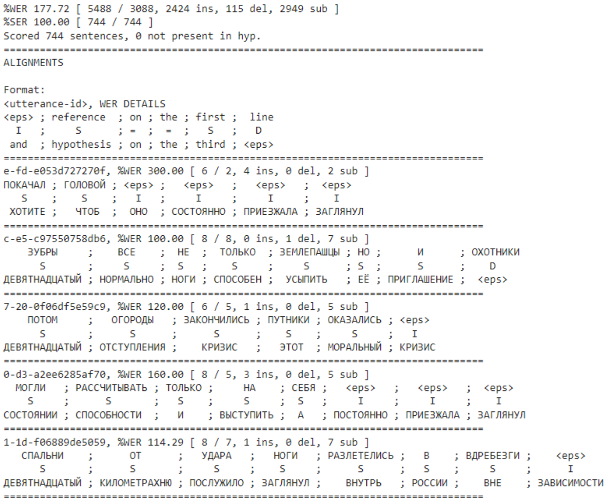


Figure 3: Experiment 02 Results

Some results for model inference - after 12 epochs training - at figure below Fig. 4.



```
%WER 177.72 [ 5488 / 3088, 2424 ins, 115 del, 2949 sub ]
%SER 100.00 [ 744 / 744 ]
Scored 744 sentences, 0 not present in hyp.
==========================================================================
ALIGNMENTS

Format:
<utterance-id>, WER DETAILS
<eps> ; reference  ; on ; the ; first ;  line
  I   ;     S      ; =  ;  =  ;   S   ;   D
 and  ; hypothesis ; on ; the ; third ; <eps>
==========================================================================
e-fd-e053d727270f, %WER 300.00 [ 6 / 2, 4 ins, 0 del, 2 sub ]
ПОКАЧАЛ ; ГОЛОВОЙ ; <eps> ;   <eps>   ;   <eps>   ;  <eps>
   S    ;    S    ;   I   ;     I     ;     I     ;    I
 ХОТИТЕ ;   ЧТОБ  ;  ОНО  ; СОСТОЯННО ; ПРИЕЗЖАЛА ; ЗАГЛЯНУЛ
==========================================================================
c-e5-c97550758db6, %WER 100.00 [ 8 / 8, 0 ins, 1 del, 7 sub ]
    ЗУБРЫ     ;    ВСЕ    ;  НЕ  ;  ТОЛЬКО  ; ЗЕМЛЕПАШЦЫ ; НО ;      И      ; ОХОТНИКИ
      S       ;     S     ;  S   ;    S     ;     S      ; S  ;      S      ;    D
ДЕВЯТНАДЦАТЫЙ ; НОРМАЛЬНО ; НОГИ ; СПОСОБЕН ;  УСЫПИТЬ   ; ЕЁ ; ПРИГЛАШЕНИЕ ;  <eps>
==========================================================================
7-20-0f06df5e59c9, %WER 120.00 [ 6 / 5, 1 ins, 0 del, 5 sub ]
    ПОТОМ     ;   ОГОРОДЫ   ; ЗАКОНЧИЛИСЬ ; ПУТНИКИ ; ОКАЗАЛИСЬ ; <eps>
      S       ;      S      ;      S      ;    S    ;     S     ;   I
ДЕВЯТНАДЦАТЫЙ ; ОТСТУПЛЕНИЯ ;    КРИЗИС   ;   ЭТОТ  ; МОРАЛЬНЫЙ ; КРИЗИС
==========================================================================
0-d3-a2ee6285af70, %WER 160.00 [ 8 / 5, 3 ins, 0 del, 5 sub ]
  МОГЛИ   ; РАССЧИТЫВАТЬ ; ТОЛЬКО ;     НА    ; СЕБЯ ;   <eps>   ;   <eps>   ;  <eps>
    S     ;      S       ;   S    ;     S     ;  S   ;     I     ;     I     ;    I
СОСТОЯНИИ ; СПОСОБНОСТИ  ;   И    ; ВЫСТУПИТЬ ;  А   ; ПОСТОЯННО ; ПРИЕЗЖАЛА ; ЗАГЛЯНУЛ
==========================================================================
1-1d-f06889de5059, %WER 114.29 [ 8 / 7, 1 ins, 0 del, 7 sub ]
   СПАЛЬНИ    ;      ОТ      ;   УДАРА   ;   НОГИ   ; РАЗЛЕТЕЛИСЬ ;   В    ; ВДРЕБЕЗГИ ;    <eps>
      S       ;      S       ;     S     ;    S     ;      S      ;   S    ;     S     ;      I
ДЕВЯТНАДЦАТЫЙ ; КИЛОМЕТРАХНЮ ; ПОСЛУЖИЛО ; ЗАГЛЯНУЛ ;    ВНУТРЬ   ; РОССИИ ;    ВНЕ    ; ЗАВИСИМОСТИ
==========================================================================
```

Figure 4: Some Predictions

The best indicators were achieved after the 15th epoch:
- valid CER: 1.45e+02, valid WER: 1.75e+02
but then began to deteriorate again to 300.
- Each epoch train time: 13-15 min on GPU K80/24Gb.

6.2.2 Language Model

The difficulties of training were discovered after working with the model with Transformer.
In Baseline, the model trained quickly to great results: train/valid loss = 0.
It was inspiring but it was a mistake.
Having adjusted the baseline, we came to other results:
- Epoch 20: train loss: 5.06, valid loss: 6.04, test loss: 5.77
- vs Epoch 1: train loss: 7.03, valid loss: 6.09
- train time each epoch 1min 10sec. (GPU K80/24Gb)

And this significantly rejuvenated the result in the future.

6.3 Transforner Language Model & Transformer Acoustic model & Russian dataset buriy-audiobooks-2-val

Initially, I checked and adjusted the parameters procedures for launching the model (not everything is as written in the tutorial).

6.3.1 Language Model

Results on GPU K80/24Gb:
- from epoch 1: train loss: 8.77, valid loss: 8.84,
- to epoch 60: train loss: 7.10, valid loss: 6.49,
- each epoch train time: 1min 20sec.

6.3.2 Acoustic Model

Results on GPU 1xK80/24Gb:
- from epoch 1: train loss: 1.90e+02, valid loss: 1.98e+02, valid ACC: 0.00e+00
- to epoch 11: test loss: 61.93, test ACC: 1.29e-01, test WER: 1.11e+02,
- each epoch train time: 5min vs 13-15 min. at CRDNN architecture.

Results after turning:
- epoch 30:
- train loss: 98.91, valid loss: 99.72, valid ACC: 1.44e-02, valid WER: 99.87
- test loss: 86.63, test ACC: 7.38e-02, test WER: 99.47

6.4 Transforner Language Model & Transformer Acoustic model & tiny Russian dataset buriy-audiobooks-2-val

Having found bad results, I decided to drastically reduce the amount of data in order to check the fundamental operability of the model.
To do this, I left only the same entry in the Train, Valid Test datasets.
Performed on the CPU.

6.4.1 Language Model

Results on CPU:
- from epoch 1: train loss: 8.77, valid loss: 8.84,
- at epoch 21: train loss: 2.63, valid loss: 2.55,
- to epoch 40: train loss: 3.00e-01, valid loss: 2.62e-01,
- each epoch train time: 35-40sec.

6.4.2 Acoustic Model

Results on CPU:
- from epoch 1: train loss: 76.97, valid loss: 72.97, valid ACC: 0.00e+00,
- to epoch 3:
- train loss: 22.67, valid loss: 22.66, valid ACC: 9.09e-01
- test loss: 15.72, test ACC: 1.00e+00, test WER: 0.00e+00,
- each epoch train time: 22sec vs 5min at full dataset

6.5 Baseline with inference by russian dataset buriy-audiobooks-2-val

An interesting random experiment was when, with the help of pre-trained English models, Russian audio was recognized.

Russian Russian pronunciation of the recognized English text was obviously in harmony with the Russian pronunciation of the original Russian speech.

text_predict_en = "NOT A GEESE DADDY WAS MICHAUD'S SO SUCCEEDED DWARF'S DEPART"
text_target_ru = "но другие стали возмущаться что совсем нет воздуха"

Pronunciation examples are given in the notebook.

The influence of an pretrained language model has affected here.

However, using an English acoustic model with Russian speech turns out to be impossible due to the use of another Tokenizer (russian vs english вifferent datasets).

6.6 Keras baseline, adapted for the use of the Russian dataset

Results on GPU K80x24Gb:
- from epoch 1: train loss: 1.40, val_loss: 1.38,
- to epoch 30(1st day)+36(2nd day): train loss: 0.27 - val_loss: 0.60,
- each epoch train time: 210-240sec.
- mean WER% : 115.63, mean CER%: 74.12 on test dataset(other book, same reader)
- mean WER% : 115.63, mean CER%: 74.12 on val (used in training) dataset
- SoTa results - mean WER%: 5.44, mean CER%: 4.29 on part of train dataset.

6.7 Experiments Analysis

I must say that the results obtained are far from SoTA results.

But this can be explained by the need for long training.
As stated in [6]: "It is worth saying that, unfortunately, these approaches today require a large amount of data. And the real results that are achieved by the classical approach, which requires from 200 to 500 hours of audio recordings marked up to train a good model based on End2End, will require many times, and maybe tens of times more data. Now this is the biggest problem of these approaches. But maybe things will change soon."

However, a similar example in Keras `https://keras.io/examples/audio/ctc_asr/` requires for around 50 epochs or more. Each epoch takes approximately 5-6mn using a GeForce RTX 2080 Ti 11Gb GPU. The model we trained at 50 epochs has a Word Error Rate (WER) = 16% to 17%.

Keras Transformer ASR `https://keras.io/examples/audio/transformer_asr/` - "In practice, you should train for around 100 epochs or more."

Repeating the experiment for a Russian dataset with one speaker showed better and faster results. But it was possible to achieve Sata results only on the train dataset - after 40 epochs.

7 Conclusion

As a result, work was carried out on the use of end-to-end Speech Recognition solutions based on SpeechBrain and Keras Transformer for the Russian dataset: buriy-audiobooks-2-val and Russian LibriSpeech.

Additionally, the pipeline was adjusted to save and use the generated models on HuggingFace.

Thus, the modified pipeline can be used for other Russian datasets for ASR purposes.

Despite the fact that it was not possible to achieve SotA results, good dynamics was achieved, which will allow using more powerful videocards and a parameters turning to achieve the best practical results.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N Gomez, Ł. Kaiser, and I. Polosukhin. “Attention is all you need”. In: *Proc. of NeurIPS* (2017).

[2] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. Dauphin. “Convolutional sequence to sequence learning”. In: *Proc. of ICML. PMLR* (2017).

[3] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. Soplin, J. Heymann, M. Wiesner, N. Chen, A. Renduchintala, and T. Ochiai. “ESPnet: End-to-end speech processing toolkit”. In: *Proc. of Interspeech* (2018).

[4] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli. “Fairseq: A fast, extensible toolkit for sequence modeling”. In: (2019). URL: `https://arxiv.org/abs/1904.01038.`.

[5] O. Kuchaiev, J. Li, H. Nguyen, O. Hrinchuk, R. Leary, B. Ginsburg, S. Kriman, S. Beliaev, V. Lavrukhin, J. Cook, P. Castonguay, M. Popova, J. Huang, and J. M. Cohen. “NeMo: a toolkit for building AI applications using Neural Modules”. In: (2019). URL: `https://arxiv.org/abs/1909.09577`.

[6] Никита Семенов. *End2End-подход в задачах Automatic Speech Recognition*. 2019. URL: `https://habr.com/ru/company/ru_mts/blog/468663/`.

[7] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli. “wav2vec 2.0: A framework for self-supervised learning of speech representations”. In: *Proc. of NeurIPS* (2020).

[8] Mirco Ravanelli, Titouan Parcollet, Peter Plantinga, Aku Rouhe, Samuele Cornell, Loren Lugosch, Cem Subakan, Nauman Dawalatabad, Abdelwahab Heba, Jianyuan Zhong, Ju-Chieh Chou, Sung-Lin Yeh, Szu-Wei Fu, Chien-Feng Liao, Elena Rastorgueva, François Grondin, William Aris, Hwidong Na, Yan Gao, Renato De Mori, Yoshua Bengio. “SpeechBrain: A General-Purpose Speech Toolki”. In: (2021). URL: `https://arxiv.org/abs/2106.04624`.

[9] Paperswithcode.com. *Speech Recognition on LibriSpeech test-clean*. 2021. URL: `https://paperswithcode.com/sota/speech-recognition-on-librispeech-test-clean`.

[10] S. Majumdar, J. Balam, O. Hrinchuk, V. Lavrukhin, V. Noroozi, and B. Ginsburg. “Citrinet: Closing the gap between non-autoregressive and autoregressive end-to-end models for automatic speech recognition”. In: (2021). URL: `https://arxiv.org/abs2104.01721`.

[11] ASR Progress. *WER are we?* URL: `https://github.com/syhw/wer_are_we`.