Speech2Text in JoeyNMT

Stefan Machmeier
Department of Computer Science
Heidelberg University, Germany
tz251@stud.uni-heidelberg.de

Robin Fleige
Department of Computer Science
Heidelberg University, Germany
lq250@stud.uni-heidelberg.de

Andre Meyering
Department of Computer Science
Heidelberg University, Germany
hp250@stud.uni-heidelberg.de

Abstract

In this paper we are looking into speech to text transformations using the JoeyNMT neural machine translation toolkit. We adapt JoeyNMT's existing code base to additionally handle audio files. We achieve this by including Torchaudio, the audio counterpart of the Torchtext project which is heavily used by JoeyNMT.

Nicht voll-ständig

1 Introduction

In this paper the possibility to use JoeyNMT for processing speech to text is worked out. Instead of translating text to text, we translate speech to text. For this purpose, the Mel Frequency Cepstral Coefficients (MFCC) of the audio files is used as input for JoeyNMT. As datasets we use audio files from the Common Voice project, where it is possible to filter by language, accent and quality, so that uniform data can be used.

Ist dieser Satz notwendig?

2 Related Work

Speech to text transformations are common but in context of JoeyNMT, we found that it references only one project that works with audio files. "Speech Joey" (Karimova, 2019) extends JoeyNMT with functionality for speech recognition and is written by Sariya Karimova[1], a former graduate research assistant at the Statistical NLP group of the Heidelberg University. However, the project does not run anymore due to breaking changes in the project's dependencies and missing explicit version numbers in their list of dependencies. Furthermore, after trying to install older versions of certain dependencies, we noticed that some were not available for more recent versions of Python such as 3.8 or 3.9. We still looked at the project as it was listed by JoeyNMT itself. In comparison to our project[2], Speech Joey uses "librosa"[3], whereas we use Torchaudio[4]. The underlying transformations should still be very similar but we were unable to compare our results against Speech Joey due to the issues mentioned earlier.

3 JoeyNMT

JoeyNMT is an open source project for neural machine translation (NMT) that aims to be easy to understand while implementing modern approaches. This gives beginners the possibility to quickly and easily understand the architecture and customize individual parts to experiment with the behavior. Since JoeyNMT's goal is to be as simple to understand as possible, they are guided by the 80/20 approach, which in this case means they want to enable 80JoeyNMT implements a Recurrent Neural Network(RNN) encoder-decoder using Gated Recurrent Unit (GRU) or Long Short Term Memory (LSTM) units. It also provides the possibility to use either a multi-layer perceptron or a blilinear interpolation as attention function. Also JoeyNMT supports word based, char based and also Byte Pair Encoding (BPE) based encodings. In the configuration file that JoeyNMT provides, you can adjust the hyperparameters of the Neural Network.

Transformer?

Source is 100% JoeyNMT Paper

4 Speech To Text

Grundsaetzli was zu speech2text schreiben.

[1] See https://www.cl.uni-heidelberg.de/~karimova/, last visited on 2021-09-09

[2] See https://github.com/bugwelle/cl-neural-networks, lasted visited on 2021-09-10

[3] See https://librosa.org/doc/latest/index.html, lasted visited on 2021-09-10

[4] See https://github.com/pytorch/audio, lasted visited 2021-09-10

4.1 Feature Extraction

As audio files are just binary blobs and cannot be worked with as text, we cannot work with JoeyNMT's existing feature extractions for text.

For audio processing, there are multiple common ways to represent audio signals in a way that features can be easily extracted. One common way is the waveform representation as can be seen in Figure 1a on page 3 for one of our training audio files. The figure visualizes amplitude changes over time. Its benefit lies in the audio detection, where values near zero indicate silence and larger amplitudes indicate louder sounds. It can not only be used to detect pauses between spoken words, but it is also possible to detect voiced and unvoiced sounds and even certain vowels can be detected (Comer, 1968).

Another way to visualize audio signals are Spectrograms, as can be seen in Figure 1b. The waveform is visualized as a Spectrogram and shows more details of the audio signals, or rather, it shows the spectrum of frequencies over time. Oftentimes, the Mel frequency spectrum is used (Stevens et al., 1937) as this represents the human audio perception.

However, there are further frequency transformations of audio signals that can be used to better extract features to translate spoken to written words.

According to Wang and Lawlor, "MFCC is one of more the successful methods" for speech to text systems which is "based on the human peripheral auditory system" (Wang and Lawlor, 2017). Therefore, MFCC (short for Mel Frequency Cepstral Coefficients) is another transformation that yields good results for speech recognition. Figure 1c shows the resulting spectrogram after applying MFCC. We note that there are possibilities to improve results even more, as described in (Winursito et al., 2018).

We refer to (Ittichaichareon et al., 2012) and (Singh and Rani, 2014) for a detailed description of MFCC.

4.2 Dataset

The dataset used throughout this paper is the open "Common Voice Corpus 7.0" dataset by Mozilla (Ardila et al., 2020). We have chosen the German language as that's what the authors are most familiar with, which allows us to better evaluate the dataset quality. The unprocessed dataset contains 26 GB of audio files which 1035 hours of spoken text according to "Common Voice".

After listening to few audio files, the authors decided to filter the dataset for one major reason: There were audio files that had accents that made it hard to understand even for native speakers.

Furthermore, even though the authors think that enterprise speech to text models should handle accents as well as differences between male and female voices, we decided that it would be out-of-scope for this paper. Therefore, all audio files were filtered according to these rules:

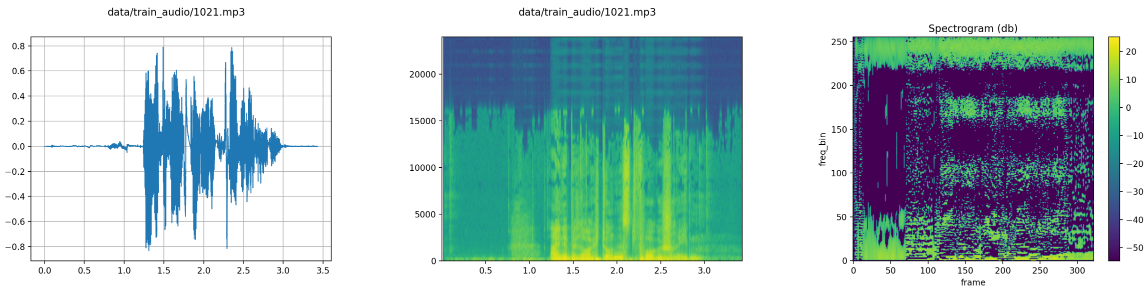
- max. 75 characters
- no accent and no Swiss- or Austrian-German
- only male voices
- no special characters in the corresponding text;
  this include punctuation such as `.,!?` and quotation marks
- lower-casing all characters in the corresponding texts

Having these limitations makes it easier to rule out issues during evaluation.

However, these limitations yield only about 7500 audio files for training and around 100 for testing and evaluation. On top of this, the quality of audio files still differs significantly. Whereas some have crystal clear voices, other have lot of noise and even contain mouse-click or keyboard sounds.

5 Results and Discussion

We compared two different models. Table 1 lists all important hyperparameters. As a basis we chose a default configuration (model A), and adapted the hidden size, and layers for decoder and encoder. As previously mentioned, we limited ourselves to MFCC only. Each model will be compared by the best validation perplexity (ppl) and Bi-Lingual Evaluation Understudy (BLEU) score that have been achieved during training. Perplexity is an exponentiation of the entropy and it tells us how well our natural language model predicts test data. A lower perplexity indicates a good prediction whereas a high perplexity indicates the opposite (Jozefowicz et al., 2016). BLEU is a quality metric score to compare machine translation. It uses the precision, and compares the appearances of the candidate, such as *love* with any other reference translation (Papineni et al., 2002).



(a) Waveform (b) Spectrogramm (c) MFCC

Figure 1: Visualizations of “Eine Beleuchtung ist vorgeschrieben.”

Table 1: Hyperparameters of both models. We only listed the most important ones. For a complete list of all hyperparameters, please refer to our GitHub repository

	Hyperparameter	**Model A**	**Model B**
	RNN type	LSTM	LSTM
	Learning rate	0.001	0.001
	level	char	char
	scheduling	plateau	plateau
	epochs	15	15
Encoder	layers	4	4
	hidden size	64	64
	dropout	0.1	0.2
Decoder	layers	4	4
	hidden size	256	512
	dropout	0.1	0.2
	hidden dropout	0.1	0.2
	attention	luong	bahdanau

Both models were trained on the same preprocessed dataset that has been described in subsection 4.2. Our idea was to have a basis that we try to improve by fine-tuning our hyperparameters. Specifically, we adapted dropout, attention, and hidden size for model B. Table 2 shows the achieved results. The first model reached a perplexity of 1.85 with a BLEU score of 3.67. This was a surprisingly good result, and we assume that the thorough preprocessing of the “Common Voice Corpus 7.0” contributed mainly to it. Figure 2 shows a steady decrease of the training loss, same applies to Figure 4. BLEU score in Figure 3 keeps increasing after each epoch. All these indicates that we do not have overfitting yet, and our perplexity is far from a saturation. We hope that model B will converge faster within 15 epochs, so that we obtain better results. Increasing the hidden size of model B, changing the dropout, and the attention lead to a perplexity of 1.63 with a BLEU score of 3.79. We assume that the increase of the hidden size has an impact to this result. As for model A, Figure 5 shows a stead decrease of the training loss. Figure 7, and Figure 6 show a similar curve to the ones of model A. Therefore, we assume that the current results in Table 2 can be improved by training more epochs, and that for both models a saturation is expected much later.

Table 2: Results of model A and B. The best validation result during training has been used.

Model	**Perplexity**	**BLEU**
A	1.8486	3.67
B	1.6352	3.79

For future work, we would try different audio transformations such as MelSpectrogram (composition of MelScale and Spectrogram), or MelScale. However, for this report, we only focused on MFCC, so for us the influence of audio transformations would be an interesting question to answer.

References

R. Ardila, M. Branson, K. Davis, M. Henretty, M. Kohler, J. Meyer, R. Morais, L. Saunders, F. M. Tyers, and G. Weber. 2020. Common voice: A massively-multilingual speech corpus. In *Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020)*, pages 4211–4215.

D. Comer. 1968. The use of waveform asymmetry to identify voiced sounds. *IEEE Transactions on Audio and Electroacoustics*, 16(4):500–506.

Chadawan Ittichaichareon, Siwat Suksri, and Thaweesak Yingthawornsuk. 2012. Speech recognition using mfcc. In *International conference on computer graphics, simulation and modeling*, pages 135–138.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling.

Sariya Karimova. 2019. Speech joey.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA. Association for Computational Linguistics.

Parwinder Pal Singh and Pushpa Rani. 2014. An approach to extract feature using mfcc. *IOSR Journal of Engineering*, 4(8):21–25.

S. S. Stevens, J. Volkmann, and E. B. Newman. 1937. A scale for the measurement of the psychological magnitude pitch. In *The Journal of the Acoustical Society of America*, pages 185–190.

Yi Wang and Bob Lawlor. 2017. Speaker recognition based on mfcc and bp neural networks. In *2017 28th Irish Signals and Systems Conference (ISSC)*, pages 1–4.

Anggun Winursito, Risanuri Hidayat, and Agus Bejo. 2018. Improvement of mfcc feature extraction accuracy using pca in indonesian speech recognition. In *2018 International Conference on Information and Communications Technology (ICOIACT)*, pages 379–383.

A Appendices

In addition, we have added a plot for loss, perplexity (ppl), and BLEU score for each model.

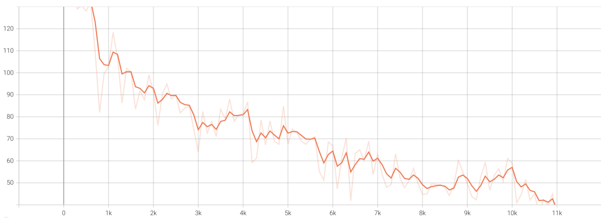


Figure 2: Model A: training loss over 15 epochs. The orange line is the loss for the training set.

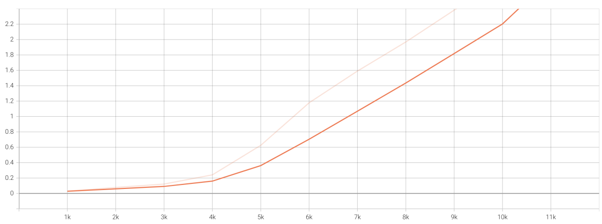


Figure 3: Model A: training BLEU score over 15 epochs. The orange line is the BLEU score for the test set.

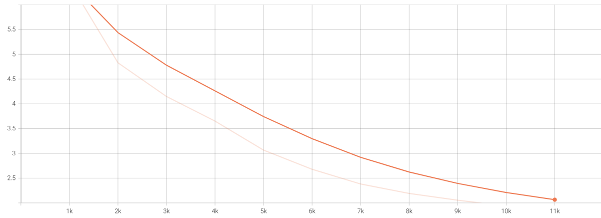


Figure 4: Model A: training perplexity (ppl) over 15 epochs. The orange line is the perplexity for the test set.

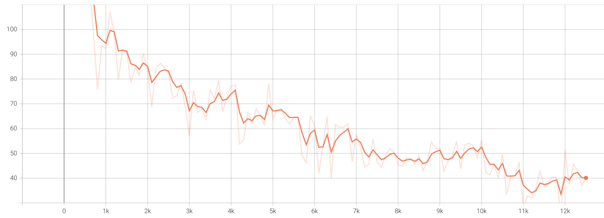


Figure 5: Model B: training loss over 15 epochs. The orange line is the loss for the training set.

B Supplemental Material

Our code is publicly available at GitHub: `https://github.com/bugwelle/cl-neural-networks`

In addition, all trained models are available at least until the end of October:

1. Model A: `https://drive.google.com/file/d/1-B5UHHfXEVUBuHNghQMrRHq1QZyXjXgK/view?usp=sharing`
2. Model B: `https://drive.google.com/file/d/1-Jz9Vn62Z18FqmHy_KusGXOIvWQbLiCq/view?usp=sharing`

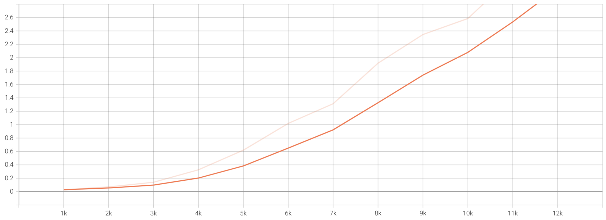


Figure 6: Model B: training BLEU score over 15 epochs. The orange line is the BLEU score for the test set.

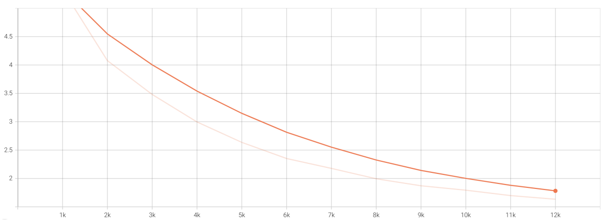


Figure 7: Model B: training perplexity (ppl) over 15 epochs. The orange line is the perplexity for the test set.