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 that are then translated to text. We achieve this by including Torchaudio, the audio counterpart of the Torchtext project which is heavily used by JoeyNMT. Even a small dataset shows promising results.

1 Introduction

In this paper an approach 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 dataset we use audio files from the Common Voice project for the German language.

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 80% of the functionality with 20% of the code. JoeyNMT 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 bilinear interpolation as attention function. Also JoeyNMT supports word based, character 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. (Kreutzer et al., 2019)

4 Speech To Text

Speech to Text is a process that converts speech in audio form to text. This is used in many different areas today, for example in the generation of subtitles for accessibility or for digital assistants such

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

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

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

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

as Alexa, Siri, Cortana or Google Home.

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:

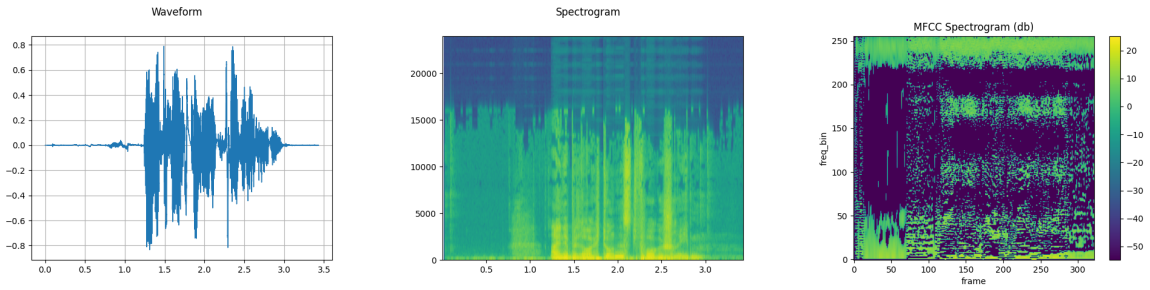
- max. 75 characters
- no accent and no Swiss- or Austrian-German
- only male voices
- no special characters in the corresponding texts;
  this include punctuation such as `.,!?` and quotation marks
- lower-casing all characters in the corresponding texts

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

However, these limitations yield only about 8300 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. Moreover, many audio files that we checked still had accents that were not part of the audio's metadata.

5 Results and Discussion

The result of our work is a model that extracts text from audio files. To see the concrete results, the translate function of JoeyNMT can be used with an audio file as input. The result of this translate function is reasonably good for the resources we had available. Hardly any of the sentences is understood completely correctly, but the output text often sounds similar to the input and the output consists almost exclusively of correct words. Therefore, we can assume that our resulting model works and that the validity would increase significantly with a larger data set as well as more epochs. For example, the audio file with the content "wir sollten



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

Figure 1: Visualizations of "Eine Beleuchtung ist vorgeschrieben."

uns nicht auf ihn verlassen" is translated to "wir sollten uns nicht auf den ersten weltkrieg der lasten" when using one of our models. This sentence shows strengths and weaknesses of our speech to text program. First of all, the beginning is translated correctly as "wir sollten uns nicht" is spoken without any accent or noise in the audio file. The end, that is, "verlassen" and "der lasten", sound phonetically similar if spoken fast and without emphasis on the *d* and *t*. However, the part in the middle is incorrect and we didn't find an explanation for it. There is no sound in the audio file that could be interpreted as "ersten weltkrieg" if a person listens to it. We noticed a similar trend in some other sentences as well where the start and end of a sentence is translated correctly but the middle part contains words that were not spoken.

We compared three 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).

All 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. For model C, we increased the hidden size even further. Table 2 shows the achieved results. The first model reached a perplexity of 1.57 with a BLEU score of 8.78. 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 2a shows a steady decrease of the training loss, same applies to Figure 2c. BLEU score in Figure 2b keeps increasing after each epoch. All this indicates that we do not have overfitting yet, and our perplexity is far from 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 7.46. We assume that the increase of the hidden size has an impact to this result. As for model A, Figure 3a shows a steady decrease of the training loss. Figure 3c and Figure 3b show a similar curve to the ones of model A. To backup our assumption, we trained an extra model C with a hidden size of 1024. Besides the 12 hours of training for 22 epochs, the results were decent for the rather limited amount of epochs. We achieved a perplexity of 1.3 with a BLEU score of 11.70. See Figure 4 in comparison to Figure 2 and Figure 3. Therefore, we assume that the current results in Table 2 can be improved by training more epochs, and that for all three models a saturation is expected much later.

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.

Table 1: Hyperparameters of all 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	Model C
	RNN type	LSTM	LSTM	LSTM
	Learning rate	0.001	0.001	0.001
	level	CHAR	CHAR	CHAR
	scheduling	PLATEAU	PLATEAU	PLATEAU
	epochs	15	15	22
Encoder	layers	4	4	4
	hidden size	64	64	64
	dropout	0.1	0.2	0.2
Decoder	layers	4	4	4
	hidden size	256	512	1024
	dropout	0.1	0.2	0.2
	hidden dropout	0.1	0.2	0.2
	attention	LUONG	BAHDANAU	BAHDANAU

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

Model	Perplexity	BLEU
A	1.5715	8.78
B	1.5214	7.46
C	1.3049	11.70

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.

Julia Kreutzer, Jasmijn Bastings, and Stefan Riezler. 2019. Joey NMT: A minimalist NMT toolkit for novices. *CoRR*, abs/1907.12484.

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 Supplemental Material

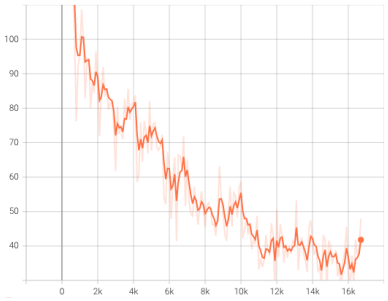
Our code is publicly available at GitHub: https://github.com/bugwelle/cl-neural-networks

In addition, all trained models are available until the end of October 2021:

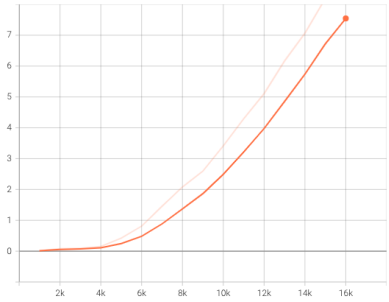
1. Model A:
   https://drive.google.com/file/d/1-IH2Kq502HC9cbv9Cgp7pGQtxg14amPA/view?usp=sharing
2. Model B:
   https://drive.google.com/file/d/1-LIxgsjLxbjxD3EbmNirjk09tms_yYEY/view?usp=sharing
3. Model C:
   https://drive.google.com/file/d/1fPK2DCX17DsMa_XmsV8QEw8FhS9swXuw/view?usp=sharing

B Appendices

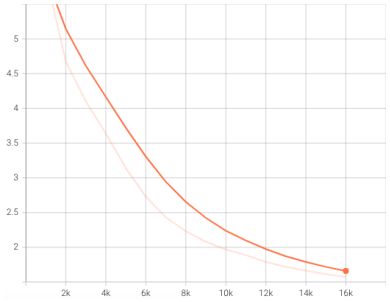
In addition, we have added a plot for loss, perplexity (ppl), and BLEU score for each model.



(a) Training loss over 15 epochs. The orange line is the loss for the training set.

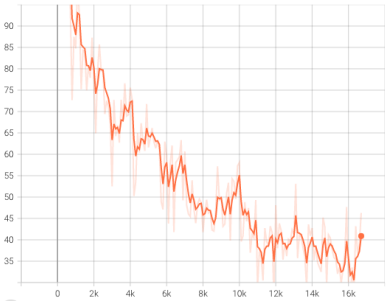


(b) Training BLEU score over 15 epochs. The orange line is the BLEU score for the test set.

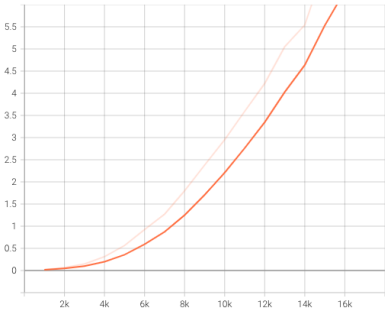


(c) Training perplexity (ppl) over 15 epochs. The orange line is the perplexity for the test set.

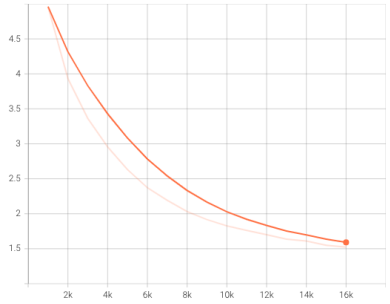
Figure 2: Model A



(a) Training loss over 15 epochs. The orange line is the loss for the training set.

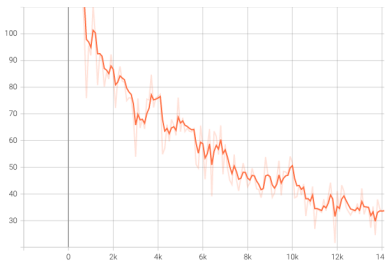


(b) Training BLEU score over 15 epochs. The orange line is the BLEU score for the test set.

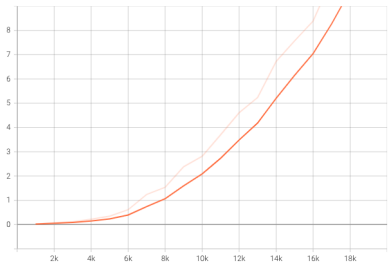


(c) Training perplexity (ppl) over 15 epochs. The orange line is the perplexity for the test set.

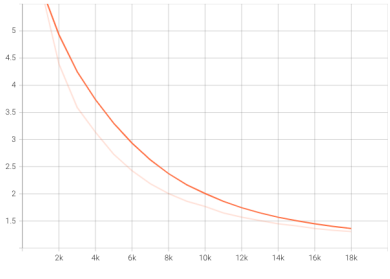
Figure 3: Model B



(a) Training loss over 22 epochs. The orange line is the loss for the training set.



(b) Training BLEU score over 22 epochs. The orange line is the BLEU score for the test set.



(c) Training perplexity (ppl) over 22 epochs. The orange line is the perplexity for the test set.

Figure 4: Model C