Speech to Text in Joey NMT

Yoalli Rezepka García
Department of Computational Linguistics
Heidelberg University, Germany
rezepkagarcia@cl.uni-heidelberg.de

David Hector
Department of Computer Science
Heidelberg University, Germany
d.hector@stud.uni-heidelberg.de

Niklas Korz
Department of Computer Science
Heidelberg University, Germany
niklas.korz@stud.uni-heidelberg.de

***Abstract*—We implement speech to text into the existing Joey NMT neural machine translation toolkit. For this, different approaches for audio processing in terms of speech recognition are analyzed and compared. The algorithm is implemented into the Joey NMT codebase by making use of Torchaudio and changing the Joey NMT implementation where necessary. Training is performed with different combinations of audio processing techniques and hyperparameters on the Interlingua dataset of the Common Voice project, showing best results with a Mel Spectrogram and decoder hidden size of 1024. An attempt to train the model with bigger datasets reveals memory related limitations in Joey NMT's use of Torchtext when combined with speech audio data.**

I. Introduction

Being able to explore a research field in an interactive and experimental manner enables beginners to quickly grasp the concepts without losing interest. With Joey NMT, there exists such an open source project for the field of neural machine translation (NMT). In this work, the possibility and details of adapting the Joey NMT project for speech to text purposes are explored. The resulting implementation is then tested against datasets from the Common Voice project, a crowd sourced speech corpus for multiple languages.

II. Joey NMT

Joey NMT is a neural machine translation toolkit that aims to be minimalist and easy to understand [1]. Its main use case is for beginners to quickly learn how to use it and how to adapt it for specific applications. Specifically, Joey NMT implements a Recurrent Neural Network (RNN) encoder-decoder and a Transformer. The RNN can be used with units of the type Gated Recurrent Unit (GRU) or Long Short Term Memory (LSTM), and a multi-layer perceptron or bilinear transformation as scoring function for its attention mechanism. The code of Joey NMT puts an emphasis on a high comment to code ratio and limits class inheritance depth to one level. To facilitate experimentation, Joey NMT parses a configuration file specified by the user for settings such as the optimizer to be used, the RNN type, attention mechanism, dataset, hidden layers, dropout rates, epochs, learning rate or batch size.

III. Speech to Text

An approach for speech to text that fits well into the architecture of Joey NMT is described by the Deep Speech 2 (DS2) paper [2]. Unlike previous techniques, it only relies on neural networks and end-to-end learning instead of acoustic models or language and pronunciation models. It is thus able to better generalize across different languages and environments, without having to invest a lot of manual work into supporting a new language. In principle, DS2 trains an RNN to receive speech spectrograms and emit text transcriptions. A training set $\{(x^{(1)}, y^{(1)}), \cdots, (x^{(n)}, y^{(n)})\}$ consists of n spectrograms of power normalized audio clips $x^{(i)}$ as features and their corresponding transcriptions $y^{(i)}$ as labels. For each time step t of a spectrogram x, the DS2 network predicts the character probability $p(\ell_t|x)$ where ℓ_t is either a character of the language's alphabet or the blank character.

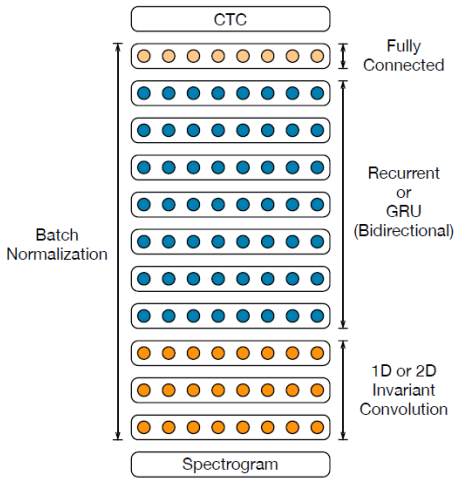

Fig. 1. Deep Speech 2 Model
Source: Amodei, Anubhai, Battenberg, *et al.* [2]

For the DS2 model's architecture, several layers of hidden units are used (fig. 1): one or more convolutional layers at the beginning, one or more recurrent layers in the middle and one or more fully connected layers at the end. In the output layer L, softmax is used to compute the probability distribution of a character at time t (eq. (1)), where h_t^{L-1} are the output activations of the last hidden layer before the output layer and k represents the defined characters.

$$p(\ell_t = k|x) = \frac{\exp(w_k^L \cdot h_t^{L-1})}{\sum_j \exp(w_j^L \cdot h_t^{L-1})} \tag{1}$$

Audio data usually come in the form of audio waves

sampled over time instead of spectrograms. Fayek [3] recommends using Mel-frequency Cepstral Coefficients (MFCCs) as spectrograms if the machine learning algorithm is prone to correlated input. The process of obtaining the MFCCs from the audio samples is described as follows [3]. A first order pre-emphasis filter $y(t) = x(t) - \alpha \cdot x(t-1)$ is applied to the signal x first, where α is a filter coefficient between 0.95 and 0.97. Then, the signal is split into short time frames of 20 to 40 milliseconds so the subsequent Fourier transformation isn't applied to the entire signal at once. Each frame is multiplied with a Hamming window (eq. (2)) where N is the window length and $0 \leq n \leq N-1$.

$$w(n) = 0.54 - 0.46 \cdot \cos\left(\frac{2\pi \cdot n}{N-1}\right) \tag{2}$$

The Hamming window is necessary to reduce spectral leakage as Fast Fourier Transformations (FFT) assume the data to be infinite. Next, the N-point FFT can be performed per frame to obtain the frequency spectrum from which the power spectrum can be computed. From these periodograms, the filter banks can be computed by applying triangular filters to extract the frequency bands. These filters make use of the Mel-scale which mimics the non-linear human ear perception of sound. A Discrete Cosine Transform is performed on the filter bank spectrogram to decorrelate the filter bank coefficients, resulting in the MFCCs. Finally, the MFCCs are normalized to balance the spectrum and improve the signal to noise ratio by subtracting the mean coefficient from all frames (fig. 2).

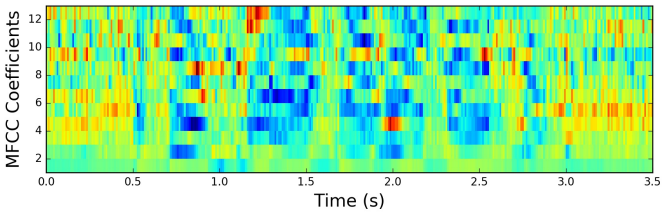

Fig. 2. Normalized Mel-frequency Cepstral Coefficients
Source: Fayek [3]

Additionally, Nguyen [4] suggests performing data augmentations on the training spectrograms to artificially increase the dataset size and prevent overfitting. The referenced spectrogram augmentation approach (SpecAugment [5]) applies time warping and frequency masking to a Log Mel Spectrogram. They found that time warping improves the model performance only minimally while being more expensive than frequency masking. In general, spectrogram augmentation helped them prevent overfitting, moreover increasing the model's performance significantly as it enabled longer training on the same data.

IV. Dataset

The Common Voice project is a collection of crowd sourced datasets for 29 different languages, containing over 2,500 hours of audio recordings and transcriptions that are corrected and validated by the community [6]. The project has been originally created by Mozilla for its DeepSpeech speech to text software. As such, it is meant to be well suited for training speech to text models. The license used for all datasets is interesting as well. The datasets are released under the terms of the CC0 "No Rights Reserved" license, which means they can practically be used as if they were public domain, thus allowing unrestricted and uncredited usage and redistribution of the data. They can be accessed at https://commonvoice.mozilla.org/. The sizes of a selection of language datasets can be seen in table I. While English is the largest Common Voice dataset with 1,469 hours of verified audio recordings and transcriptions, it still is significantly smaller than the 11,940 hours long English dataset used in the DS2 paper [2]. Nonetheless, an open dataset fits well for educational and experimental purposes.

TABLE I
Common Voice dataset sizes (September 22, 2020)

Language	Binary Size	Verified Recordings	Voices
English	50 GB	1,469 hours	61,528
German	19 GB	692 hours	11,731
Spanish	13 GB	290 hours	18,906
Esperanto	2 GB	83 hours	505
Interlingua	195 MB	5 hours	27

V. Integration

Kreutzer, Bastings, and Riezler [1] state that Joey NMT aims to "achieve 80% of the translation quality with 20% of a common toolkit's code size" by only implementing "the most common features (the bare necessities)" of recent NMT publications and implementations. Following this philosophy, this work aims to implement speech to text into the existing Joey NMT codebase by making only the most necessary changes and reusing as many parts of Joey NMT as possible. Furthermore, as the result is meant to be experimented on by novices, the implementation has to remain configurable through the user-specified configuration files. Joey NMT already makes use of the PyTorch library for machine learning and Torchtext for loading and batching text datasets. Similarly, the speech to text adaptation makes use of the Torchaudio library for loading the Common Voice datasets, decoding the audio files and performing the spectrogram transformations on the raw audio data. The configuration file structure requires only a simple change. Originally, the user specified the source and target language, as well as the dataset locations. Now, the user merely specifies the language on which speech to text is to be performed and the implementation will download and load the according Common Voice dataset. As Torchaudio also supports other datasets next to Common Voice, it would be possible to add a configuration field for the dataset source or a local dataset path to be used.

The main issue with integrating the speech to text process into the existing Joey NMT codebase is its dependency on Torchtext's dataset and batch implementations. Integrating the Torchaudio datasets into these interfaces requires loading the entire data to be used into memory and passing it to an instance of Torchtext's own dataset implementation. Furthermore, Torchtext naturally relies on having a vocabulary and indexes the passed data based on the vocabulary set. In case of audio spectrograms, there is no vocabulary and no need to index the audio data as its representation already consists of nothing but numbers. As the audio preprocessing already functions as a kind of embedding, the source embedding has been replaced with a new class that simply passes the incoming values as embeddings. On a further note, a bug in the Torchaudio Common Voice implementation prevented the downloaded dataset from being extracted into the correct path. Therefore, said implementation was forked into this work's codebase to fix it.

VI. Training and Evaluation

Training was performed on the Interlingua dataset. An attempt to train on bigger datasets such as Esperanto and Spanish was made. Before performing full training with more than 10 epochs, smaller test runs of 10 epochs were used to compare the effects of different audio preprocessing techniques and of changing the decoder's hidden size. The decoder's hidden size was set to 128 and 1024 respectively. The utilized audio preprocessing techniques are listed in table II. All remaining hyperparameter configurations used for training can be seen in table III.

TABLE II
Audio Preprocessing Techniques

Name	Emb Dim	Description
MFCC	40	Mel-frequency Cepstral Coefficients
LW Mel	128	Letter-width Mel Spectrogram
LW Spec	201	Letter-width Spectrogram
LW Mel Aug	128	LW Mel with SpecAugment

VII. Results and Discussion

The metric used for comparing the configurations is the validation perplexity (ppl). Training with MFCC as audio preprocessing technique turned out to be very slow and showed not to render any useful results. Because of this, training with MFCC was not completed and disregarded for the results. Increasing the decoder's hidden size from 128 to 1024 resulted in a significant improvement (table IV). This was also directly visible in the test output. While the initial model would only emit a limited set of repeating letters instead of words, increasing the hidden size led to real words forming as output. The letter-width Mel Spectrogram configuration with hidden size 1024 was trained for 100 epochs afterwards to see how far the model could go with such a small dataset. The outcome was that the validation perplexity was only able to improve until epoch 27 with a perplexity of 4.85, any further training leading to the over-fitting. Still, the model suffers from the fact that is was trained on no more than a subset of the 5 hour Interlingua dataset. In comparison, the English dataset used in Deep Speech 2 is 2388 times the size. Nevertheless, the model is able to generate mostly complete, different sentences per test input, even though they do not match the expected transcriptions.

TABLE III
Training Hyperparameters

Name	Value
Level	Char
Lowercase	True
Optimizer	Adam
Learning Rate	0.0002
Scheduling	Plateau
RNN Type	LSTM
Enc Hidden Size	5
Enc Bidirectional	True
Enc Dropout	0.2
Enc Num Layers	7
Dec Emb Dim	512
Dec Dropout	0.2
Dec Hidden Dropout	0.2
Dec Num Layers	4
Dec Input Feeding	True
Dec Init Hidden	Bridge
Dec Attention	Bahdanau

TABLE IV
Training Validation Results (10 epochs)

Audio Technique	Hidden Size	Perplexity
LW Mel	128	9.64
LW Mel	1024	7.26
LW Spec	1024	7.42
LW Mel Aug	1024	7.59

Trying to train on the Spanish dataset led to issues with the available memory. As explained before, converting the Torchaudio dataset to a Torchtext dataset requires loading the entire data into memory to pass it to the new instance which is impractical for larger datasets. Similarly, Joey NMT requires a large amount of memory during training when combining audio data with a sensible configuration of hyperparameters. On newer GPUs with 8 gigabytes of dedicated memory, the allocation limit was quickly reached. Thus, Esperanto training could only be performed with a smaller decoder embedding dimension and hidden size. All other configuration parameters

for the Esperanto training were taken from the best result on the Interlingua dataset. Training the model on Esperanto for 30 epochs with a smaller hidden size had the effect that, while still emitting whole words instead of repeated characters, the generated sentences were almost equal except for a few words for any input given to the model. In conclusion, Speech to Text in Joey NMT has great potential but is held back by the memory restrictions of its current implementation. Removing Torchtext from the Joey NMT codebase and replacing it with an approach that allows for loading and processing the Torchaudio data per batch may permit training on a larger dataset such as the Common Voice English dataset and in turn lead to accurate transcriptions.

REFERENCES

[1] J. Kreutzer, J. Bastings, and S. Riezler, "Joey NMT: A Minimalist NMT Toolkit for Novices," Jun. 18, 2020. arXiv: 1907.12484 [cs].

[2] D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, E. Elsen, J. Engel, L. Fan, C. Fougner, T. Han, A. Hannun, B. Jun, P. LeGresley, L. Lin, S. Narang, A. Ng, S. Ozair, R. Prenger, J. Raiman, S. Satheesh, D. Seetapun, S. Sengupta, Y. Wang, Z. Wang, C. Wang, B. Xiao, D. Yogatama, J. Zhan, and Z. Zhu, "Deep Speech 2: End-to-End Speech Recognition in English and Mandarin," Dec. 8, 2015. arXiv: 1512.02595 [cs].

[3] H. Fayek. (Apr. 21, 2016). Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What's In-Between, [Online]. Available: https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html (visited on 08/24/2020).

[4] M. Nguyen. (May 18, 2020). Building an end-to-end Speech Recognition model in PyTorch with AssemblyAI, [Online]. Available: https://www.comet.ml/site/customer-case-study-building-an-end-to-end-speech-recognition-model-in-pytorch-with-assemblyai/ (visited on 08/20/2020).

[5] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition," *Interspeech 2019*, pp. 2613–2617, Sep. 15, 2019. DOI: 10.21437/Interspeech.2019-2680. arXiv: 1904.08779.

[6] R. Ardila, M. Branson, K. Davis, M. Henretty, M. Kohler, J. Meyer, R. Morais, L. Saunders, F. M. Tyers, and G. Weber, "Common Voice: A Massively-Multilingual Speech Corpus," Mar. 5, 2020. arXiv: 1912.06670 [cs].