Automatic Speech Recognition for South African Languages

UNIVERSITEIT
iYUNIVESITHI
STELLENBOSCH
UNIVERSITY

Lucas Meyer

Research assignment presented in the partial fulfilment
of the requirement for the degree of
MSc (Machine Learning & Artificial Intelligence)
at the University of Stellenbosch

Supervisor: Prof. H. Kamper

Degree of confidentiality: A November 2023

PLAGIARISM DECLARATION

1. Plagiarism is the use of ideas, material and other intellectual property of another's work and to present it as my own.
2. I agree that plagiarism is a punishable offence because it constitutes theft.
3. Accordingly, all quotations and contributions from any source whatsoever (including the internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism.
4. I also understand that direct translations are plagiarism.
5. I declare that the work contained in this research assignment, except otherwise stated, is my original work and that I have not previously (in its entirety or in part) submitted it for grading in this research assignment or another research assignment.

22614524	[illegible]
Student number	Signature
L. Meyer	6 November 2023
Initials and surname	Date

ACKNOWLEDGEMENTS

The Department of Statistics and Actuarial Science (the Department) wishes to acknowledge David Rodwell for generously creating a template based off the USB (University of Stellenbosch Business School) guidelines which have been adapted for the purposes of the department.

ABSTRACT

Insert an abstract of not more than 500 words here

Key words:

Technical guidelines; Chapter outline; Examples

CHAPTER 1

INTRODUCTION

Enter introduction here ...

CHAPTER 2

BACKGROUND

2.1 THE AUTOMATIC SPEECH RECOGNITION TASK

Speech recognition or automatic speech recognition (ASR) is the task of predicting the text, sentence, transcription, or sequence of characters for a corresponding speech recording. The general approach of ASR is to first compute feature representations called *speech features* from given audio data, and then map the speech features to characters or other tokens such as wordpieces.

2.1.1 Speech recognition data

The first step of creating an ASR model is to prepare the data that is used for the model. A single data entry for an ASR dataset is a speech recording (typically in waveform audio file format) along with a corresponding text transcription of the words which are spoken. We explain why the choice of dataset has a significant effect on the accuracy of ASR models.

The amount of training data: The more data that is available during training the better the ability of the ASR model to generalize. A small dataset (with few unique voices) may lead to overfitting to the specific voices in the dataset.

Read speech vs. conversational speech: Humans tend to pronounce their speech more clearly when reading text from a transcript, and recent ASR models can predict read speech very accurately [1]. In contrast, accurately predicting conversational speech is still a major challenge in ASR.

Tonal qualities for different accents: The accent of the speaker, which depends on the gender, age and ethnicity of the speaker is another important factor to bare in mind. Generally, male speakers have a lower pitch compared to female speakers. Similarly, adult voices are generally have a lower pitch compared to children.

The audio quality of speech recordings: The position of the microphone, the quality of the microphone, the number of microphones available, and the presence of background noise contribute towards the quality of speech recordings.

2.1.2 Speech features

Computing speech features is useful because audio data consists of a one-dimensional array of integers that describe the amplitude of the recorded sound wave for small time periods called *samples* (see Figure **??**). The issue is that mapping a sequence of amplitude measurements to a sequence of characters is impractical. A common technique used to compute speech features is to transform the audio data from the amplitude-time domain to the frequency-time domain, using the Fast Fourier Transform (FFT) algorithm [2], [3]. However, in this study we discuss a more recent feature extraction approach based on contrastive learning.

2.2 WAV2VEC 2.0

wav2vec 2.0 provides a framework for learning speech representations using unlabeled speech data. The wav2vec 2.0 architecture is described by the network diagram in Figure 2.1 There are four important components of the wav2vec 2.0 architecture: the feature-encoder, the context network, the quantization module, and the objective function.

The basic idea is that the quantization module contains vectors called codebook entries. The goal is to match each small segment of the input audio with a codebook entry. The codebook entries are concatenated to create speech features and used for Speech Recognition and other tasks such as Speech Translation and Speech Classification.

2.2.1 Feature encoder

The feature encoder maps the raw audio data (speech recordings) to latent speech representations: $f : \mathcal{X} \to \mathcal{Z}$. Thus, the feature encoder f maps a sequence of audio samples $\mathbf{x}^{(1)}, \ldots \mathbf{x}^{(N)}$ into a sequence of latent feature vectors $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(t)}$.

The audio data is scaled to have zero mean and unit variance before going into the feature encoder. The feature encoder consists of seven convolutional blocks, where each convolutional block contains a temporal[1] convolutional layer, a layer normalization layer, and the GELU activation function.

[1]One-dimensional convolutional layer designed for sequential data.

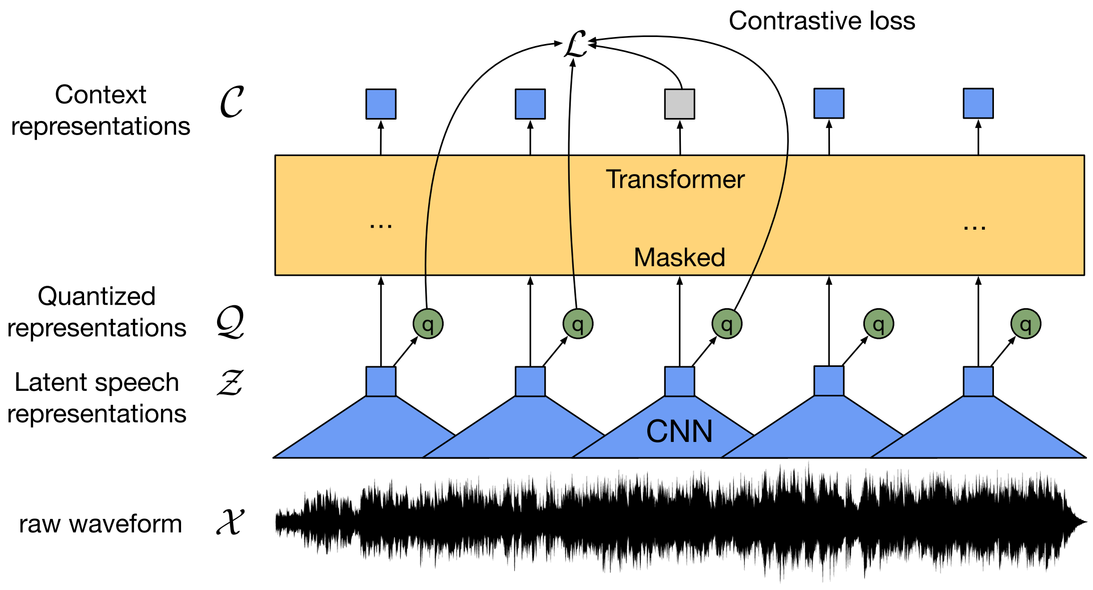

Figure 2.1: A visualization of the network architecture of wav2vec 2.0, taken from the original paper [4].

Each temporal convolutional layer contains 512 channels. The strides of the seven temporal convolutional layers are $(5, 2, 2, 2, 2, 2, 2)$ and the kernel widths are $(10, 3, 3, 3, 3, 2, 2)$. The strides used results in each $\mathbf{z}^{(t)}$ representing 25ms of audio (or 400 input samples), strided by about 20ms.

Layer normalization scales the logits after each convolutional layer to have zero mean and unit variance, which has shown to increase the chances of earlier convergence. GELU has become a popular activation function for NLP related tasks

2.2.2 Quantization module

The quantization module maps the latent speech features into discrete speech units: $h : \mathcal{Z} \rightarrow \mathcal{Q}$. Speech is sound, and sound is represented as a continuous function. We would like to use Transformers and so continuous representations will not work. Unlike written language, which can be discretized into tokens such as characters or sub-words, speech does not have natural subunits [5]. The quantization module is a method in which discrete speech units are automatically learned using product quantization.

To perform product quantization, the quantization module uses G *codebooks*, where each codebook contains V *codebook entries* $\mathbf{e}_1, \ldots, \mathbf{e}_V$.

The following steps describe the process of automatically assigning a discrete speech unit to each latent speech feature $\mathbf{z}^{(t)}$:

1. Transform $\mathbf{z}^{(t)}$ into $\mathbf{l}^{(t)} \in \mathbb{R}^{G \times V}$ using a linear transformation.
2. Choose one codebook entry $\mathbf{e}_g$ for each codebook $g = 1, \ldots, G$, based on the values of $\mathbf{l}^{(t)}$.
3. Concatenate the codebook entries $\mathbf{e}_1, \ldots, \mathbf{e}_G$.
4. Transform the resulting vector into $\mathbf{q}^{(t)} \in \mathbb{R}^f$ using another linear transformation.

The two linear transformations are feed-forward neural networks $\mathrm{FF}_1 : \mathbb{R}^f \to \mathbb{R}^{G \times V}$ and $\mathrm{FF}_2 : \mathbb{R}^d \to \mathbb{R}^f$. In the second step above, the codebook entry $\mathbf{e}_g$ is chosen as the one with the argmax of the logits $\mathbf{l}$. Choosing the codebook entries in this way is non-differentiable. Fortunately, we can use the Gumbel softmax to choose codebook entries in a fully differentiable way. $\mathbf{e}_g$ is chosen as the entry that maximizes

$$p_{g,v} = \frac{\exp\left(\mathbf{l}^{(t)}_{g,v} + n_v\right)/\tau}{\sum\limits_{k=1}^{V} \exp\left(\mathbf{l}^{(t)}_{g,k} + n_k\right)/\tau}, \tag{2.1}$$

where τ is a non-negative temperature, $n = -\log(-\log(u))$, and u are uniform samples from $\mathcal{U}(0,1)$. During the forward pass, codeword i is chosen by $i = \mathrm{argmax}_j p_{g,j}$ and in the backward pass, the true gradient of the Gumbel softmax outputs is used.

2.2.3 Context network

The context network, which follows the Transformer architecture, maps the latent speech features into discrete speech units: $h : \mathcal{Z} \to \mathcal{Q}$.

2.2.4 Objective function

2.3 CONNECTIONIST TEMPORAL CLASSIFICATION

Connectionist Temporal Classification (CTC) [6] is an algorithm (or loss function) developed to map a sequence of speech features to a sequence of characters. The authors of the wav2vec 2.0 paper suggest that if finetuning wav2vec 2.0 for ASR one should add a linear layer and CTC on top of the wav2vec 2.0 network after pretraining on unlabeled audio data.

2.3.1 Decoding with CTC

2.3.2 Improving performance with a language model

2.4 FINETUNING PRETRAINED WAV2VEC 2.0 MODELS

Pretraining on unlabeled audio data with wav2vec 2.0 is expensive and inconsistent.

2.4.1 XLS-R

CHAPTER 3

EXPERIMENTAL SETUP

3.1 DATA SETS

We collected three datasets from which we created the datasets used for pretraining and finetuning. Note that the datasets for pretraining and finetuning are mutually exclusive. We describe the three datasets in the following paragraphs.

NCHLT dataset Majority is used for pre-training, the rest is used for fine-tuning.

FLEURS dataset Used for fine-tuning

High Quality TTS dataset Used for fine-tuning

3.2 PRE-TRAINING

3.3 FINE-TUNING

3.4 EVALUATION METRICS

Word error rate The word error rate (WER) is equal to the number of character-level errors in the predicted transcript, divided by the number of words in the true transcript. One character-level error is corrected using one of three operations: inserting a new character, deleting an existing character, or substituting an existing character for a new character.

CTC Score/loss ...

CHAPTER 4

RESULTS

CHAPTER 5

CONCLUSIONS

REFERENCES

[1] Daniel Jurafsky and James H Martin. Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition.

[2] William T Cochran, James W Cooley, David L Favin, Howard D Helms, Reginald A Kaenel, William W Lang, George C Maling, David E Nelson, Charles M Rader, and Peter D Welch. What is the fast fourier transform? *Proceedings of the IEEE*, 55(10):1664–1674, 1967.

[3] James W Cooley, Peter AW Lewis, and Peter D Welch. The fast fourier transform and its applications. *IEEE Transactions on Education*, 12(1):27–34, 1969.

[4] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations, 2020.

[5] Jonathan Bgn. An illustrated tour of wav2vec 2.0, 2021.

[6] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376, 2006.