Gabriel Marcelino, Grant Burk

CST-440

January 24, 2025

CLC – Voice Recognition Systems

Comparison of a Custom Speech-to-Text Model with a Pretrained Model

Introduction

In this project, we aim to compare the effectiveness of a custom RNN model with a pretrained Wav2Vec2 model for transcription. We will train the model using the Commom Voice dataset, with features extracted as Mel-Frequency Cepstral Coefficients (MFCCs) and tokenized transcriptions. We aim to to explore different setups and to achieve a word-level accuracy of approximately 33%(Model predicts the right word roughly once every 3 words). Our findings highlight the challenges of training a custom model from scratch with limited time and resources, contrasted against the efficiency and accuracy of the pretrained Wav2Vec2 model.
See code at https://github.com/gabrielrosendo/speech-recognition/blob/main/SpeechRecognition.ipynb

2. Data Preparation

The dataset used for this study Mozilla's Common Voice Delta Segment 20.0 dataset, available at https://commonvoice.mozilla.org/en/datasets. Audio files were resampled to 16 kHz as a standard. For feature extraction, Mel-Frequency Cepstral Coefficients (MFCCs) were computed using 13 coefficients, a hop length of 512, and a window length of 2048. Transcriptions were tokenized at the character level, and sequences padded or truncated to a fixed length.

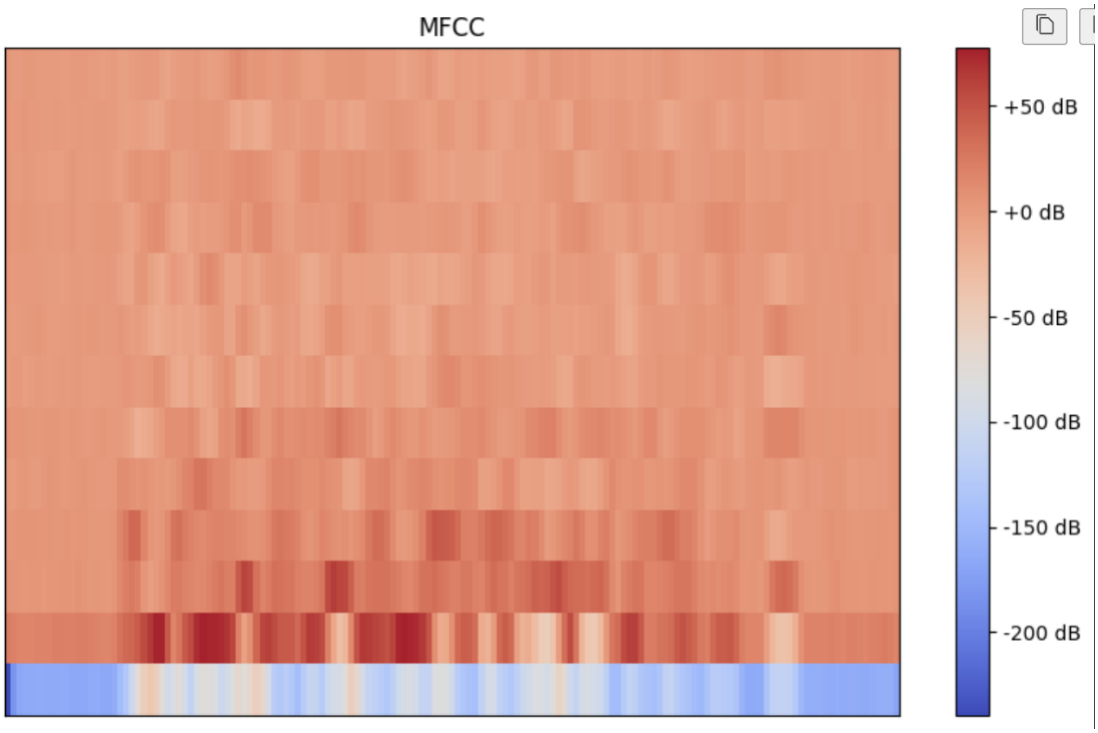

Figure 1: MFCC of an audio sample after feature extraction

3. Models Used

We used two approaches for the transcription task: a custom LSTM-based model and the pretrained Wav2Vec2 model. Our initial setup for the custom model was a Bidirectional LSTM layer with 128 units, a dropout layer (rate 0.2) to mitigate overfitting, and a time-distributed dense layer with softmax activation for character-level predictions. The model was trained using the Adam optimizer and sparse categorical crossentropy loss for 30 epochs, with early stopping. We will adjust the hyperparameters for this according to the results.

The pretrained Wav2Vec2 model, "facebook/wav2vec2-base-960h," was used for comparison. This model, developed by Facebook AI, was fine-tuned on 960 hours of labeled speech data and can generate accurate transcriptions.

---

4. Experiments and Results

Initial Results

During training with the initial setup, I already noticed something odd. By the second epoch, it was claiming it already had a accuracy of almost 80%, which suggested that something might have been wrong with the setup. This rapid surge in training and validation accuracy seemed indicative of potential overfitting or a fundamental issue with the way the model was learning.

```
Epoch 1/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 88s 14s/step - accuracy: 0.4565 - loss: 3.5676 - val_accuracy: 0.7585 - val_loss: 2.9865
Epoch 2/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 30s 6s/step - accuracy: 0.7950 - loss: 2.8310 - val_accuracy: 0.8266 - val_loss: 1.5681
Epoch 3/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 35s 7s/step - accuracy: 0.8453 - loss: 1.1820 - val_accuracy: 0.8266 - val_loss: 0.8564
Epoch 4/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 31s 7s/step - accuracy: 0.8397 - loss: 0.7986 - val_accuracy: 0.8266 - val_loss: 0.8334
Epoch 5/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 21s 4s/step - accuracy: 0.8354 - loss: 0.7867 - val_accuracy: 0.8266 - val_loss: 0.7824
Epoch 6/30
5/5 ━━━━━━━━━━━━━━━━━━━━ 23s 5s/step - accuracy: 0.8411 - loss: 0.7243 - val_accuracy: 0.8266 - val_loss: 0.7481
```

Figure 2: Epochs 1–6 displaying a rapid surge in training accuracy, with validation accuracy stabilizing early, indicating potential overfitting

This was confirmed by the F1-scores and precision-recall metrics, which were extremely low across all classes, indicating poor performance on actual predictions.

	precision	recall	f1-score	support
0	0.00	0.00	0.00	0
1	0.16	0.62	0.26	459
2	0.06	0.02	0.03	309
3	0.00	0.00	0.00	215
4	0.00	0.00	0.00	227
5	0.00	0.00	0.00	202
6	0.00	0.00	0.00	179
7	0.14	0.01	0.01	180
8	0.00	0.00	0.00	176
9	0.00	0.00	0.00	160
10	0.00	0.00	0.00	128
11	0.00	0.00	0.00	111
12	0.00	0.00	0.00	100
13	0.00	0.00	0.00	63
14	0.00	0.00	0.00	69
15	0.00	0.00	0.00	66
16	0.00	0.00	0.00	45
17	0.00	0.00	0.00	48
18	0.00	0.00	0.00	55
19	0.00	0.00	0.00	49
20	0.00	0.00	0.00	28
21	0.00	0.00	0.00	45
...				
accuracy			0.10	3038
macro avg	0.01	0.02	0.01	3038
weighted avg	0.04	0.10	0.04	3038

Figure 3: The results indicate poor model performance across all classes despite high validation accuracy

This was further confirmed by the actual predictions, consisting almost entirely of zero values, as shown below. This confirmed once again that the the model had not learned meaningful patterns.

```
  [ 0.0000000e+00  0.0000000e+00  0.0000000e+00 ...  0.0000000e+00
    0.0000000e+00  0.0000000e+00]
  [ 0.0000000e+00  0.0000000e+00  0.0000000e+00 ...  0.0000000e+00
    0.0000000e+00  0.0000000e+00]
  [ 0.0000000e+00  0.0000000e+00  0.0000000e+00 ...  0.0000000e+00
    0.0000000e+00  0.0000000e+00]]]
Prediction shape: (1, 375, 37)
Decoded Prediction:                                t t            t t
```

Figure 4: Example of nonsensical predictions outputted by the model, with the majority of values being zeros and decoded predictions showing meaningless sequences ('t t t t').

Causes and solutions

The primary issue with the model's poor performance stems from two key factors: insufficient data quantity and lack of data diversity during training.

Insufficient Data Quantity

Training an AI model requires a robust dataset that adequately represents the problem domain. In this case, the dataset used under the title "delta" likely contained too few examples

for the model to generalize effectively. A dataset with limited examples constrains the model's ability to learn the full range of patterns or features required to perform well on unseen data. While the model may perform well during training (achieving an accuracy of ~0.85), this can often result in overfitting—the model essentially memorizes the training data rather than learning generalizable patterns. This discrepancy between training accuracy and test performance highlights that the model has not generalized effectively to new, unseen examples.

Proposed Solution

The solution lies in addressing both the quantity and diversity of the dataset:

Increase Dataset Size

Collect or source a larger dataset with significantly more examples. A larger dataset allows the model to observe a broader range of patterns and features, helping it develop a more nuanced understanding of the task. For speech data specifically, this would involve gathering recordings with a variety of:

1. Accents
2. Speaking speeds
3. Background noises
4. Audio qualities (e.g., microphone differences)

Ensure Dataset Diversity

Beyond increasing the dataset size, it is critical to ensure that the data reflects the distribution of the test data. If the test data includes speech from a variety of demographics, environments, or scenarios, the training data must also represent those conditions. This alignment reduces the risk of encountering entirely unfamiliar patterns during testing.

Extend Training Duration

Training on a larger dataset would require more epochs or iterations for the model to fully learn the underlying patterns. However, this must be done cautiously to avoid overfitting. Regular monitoring of training and validation loss can ensure that the model is learning effectively without memorizing the training data.

Data Augmentation

If collecting new data is challenging, data augmentation techniques can artificially increase dataset size and diversity. For speech data, this could include:

1. Adding noise to audio files
2. Speed or pitch modulation
3. Time stretching or compression
4. Adding reverberation effects
5. Evaluate with Cross-Validation

Some of these techniques were attempted in this project but to no avail.

Broader Implications

By using a more extensive and varied dataset, the model will be better equipped to handle real-world scenarios, where input data is often unpredictable and noisy. Generalization is key for any AI model to perform well outside the controlled environment of its training data. Adopting these strategies will not only improve test accuracy but also ensure the model is robust and scalable for practical applications.

Wav2Vec2 Model Results and Analysis

- WER: 0.29, CER: 0.06
- Average Word Error Rate (WER) for subset: 0.32
- Average Character Error Rate (CER) for subset: 0.09

The pre-trained Wav2Vec2 model demonstrated strong performance, achieving a Word Error Rate (WER) of 0.29 and a Character Error Rate (CER) of 0.06. These results indicate relatively low error rates, with the model making only minor mistakes. The subset evaluation further reinforces this performance, with an average WER of 0.32 and an average CER of 0.09, showing consistency across different samples. These metrics suggest that the pre-trained model is effective at transcribing speech, with minimal word- and character-level inaccuracies, making it a solid baseline.

---

4. Discussion

Insights from the Comparison

Pretrained models, like Wav2Vec2, offer significant advantages in terms of resource efficiency and accuracy. With a Word Error Rate (WER) of 0.29 and Character Error Rate (CER) of 0.06, Wav2Vec2 shows strong performance with minimal preprocessing required. In contrast, the custom model struggled due to limited data, overfitting, and lack of data diversity, hindering its ability to generalize well.

Advantages of Pretrained Models for Resource Efficiency and Accuracy

Pretrained models significantly reduce the computational burden as they leverage transfer learning, allowing developers to use models that are already optimized for certain tasks. These models require fewer training resources, achieve higher accuracy out of the box, and can be fine-tuned with less data compared to training a custom model from scratch. Wav2Vec2 exemplifies this by providing strong baseline performance even without extensive fine-tuning.

Challenges of Training Custom Models

Custom models face difficulties with insufficient and non-diverse training data, leading to overfitting and poor generalization. The "delta" dataset structure, which lacked overlap with test data, exacerbated these issues, resulting in poor performance on unseen data.

Suggestions for Improving the Custom Model

To improve the custom model, increasing dataset size and diversity is crucial. Data augmentation techniques can help expand the dataset, while regularization methods like dropout can reduce overfitting. Cross-validation should also be used to assess generalization. However, the results underscore that pretrained models like Wav2Vec2 are often more practical and accurate, especially with limited resources.

---

5. Conclusion

This project compared the performance of a custom RNN-based speech-to-text model with a pretrained Wav2Vec2 model using Mozilla's Common Voice dataset. The custom model utilized MFCCs for feature extraction and character-level tokenization but struggled with overfitting and poor generalization. While training accuracy reached ~0.85, the test performance was poor due to insufficient data quantity and diversity. The dataset's "delta" structure, where each data point differed significantly from the rest, caused a lack of overlap between training and test sets, resulting in nonsensical predictions and low precision-recall metrics. These challenges highlighted the difficulty of training custom models from scratch with limited data and resources.

In contrast, the pretrained Wav2Vec2 model, fine-tuned on 960 hours of labeled speech data, demonstrated strong transcription accuracy with a Word Error Rate (WER) of 0.29 and a Character Error Rate (CER) of 0.06. This performance underscores the efficiency and reliability of pretrained models, which require less preprocessing and are better equipped for real-world speech recognition tasks. To improve the custom model, strategies such as increasing dataset size, employing data augmentation, and extending training duration could be employed. However, this study ultimately emphasizes the practicality of pretrained models like Wav2Vec2, which provide high accuracy and scalability without the need for extensive resources.

References

https://huggingface.co/facebook/wav2vec2-base-960h