readme.md

Subtitle-AI

Team Member

Macy So, Lazaro Solorzano, Jared Chou

Problem Statement

Closed captioning displays the audio portion of a television program as text on the TV screen, providing a critical link to news, entertainment, and information for individuals who are deaf or hard of hearing.The importance of this project is to enhance the educational learning experience for students and others with disabilities.There should be no restrictions for simply trying to watch a video or anything related to the sort. With subtitles/closed captions it will help eliminate this issue by being able to provide it to services that don't already provide their closed captions. This relates to the work that we have done in class because it is building on transformers. Which provides APIs and tools to easily download and train state-of-the-art pre-trained models. Using pre-trained models can reduce your computing costs, carbon footprint, and save you the time and resources required to train a model from scratch. With this in mind, we worked on what is considered to be an important NLP task in our case being Speech-to-Text.

Model Accuracy Measure

The metric that we will use for our project will be transcription accuracy based on the partitioned test data set, evaluated based on the word error rate of our model. Our baseline aim will be for at most an average of 10% word error rate accuracy on testing data for success, looking to push for much higher accuracy if we have time to do so.

Abstract

Without a closed caption, all this vital information will be lost by a viewer with a hearing problem. But when a program has a closed caption, a viewer with a hearing problem can pick up on sarcasm, understand the vibe of the crowd, and understand who is talking when they are not on screen. We are trying to solve the problem of providing a service for every live audio translation with closed captioning, as not every streaming service/video has closed captions. We are also aiming to increase the accuracy of closed captioning during this research process. We will be using a pre-existing model, and utilizing transfer learning to create our own model. And to test it, we will be providing some new audio data to increase precision. The library in python that we will be using to create the model is Wave2Vec. With vast amounts of fine-tuning to the Wav2Vec2 model already, changes in the hyperparameter configuration of models will result in minimal performance changes. This implies that fine-tuning on the model will have a much higher impact on the model's performance, rather than specific hyperparameters. We were able to achieve an average of 4.13% word error rate on the full set of testing data.

Background

We are utilizing wav2vec 2.0 as the basis of our research paper. wav2vec 2.0 is a pre-trained model developed by Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. This model learns powerful representations of speech audio followed by fine-tuning on transcribed speech which results in very strong performances while being conceptually easier to understand than other semi-supervised learning models.

Wav2Vec2 is a speech model that accepts a float array corresponding to the raw waveform of the speech signal, and converts it into a text transcription prediction. We are utilizing the pretrained weights to perform further hyperparameter fine tuning in order to maximize the performance of our tuned model.

In addition to this, we are utilizing the LibriSpeech dataset which is a large scale, labeled corpus of read English speech. wav2vec 2.0 was trained, fine-tuned, and tested on this dataset. We will be utilizing the clean testing dataset from LibriSpeech.

Method

We are using a pre-trained Wav2Vec model to train our data, and fine-tune hyperparameters in order to optimize the performance of the model. The Wav2Vec model is pre-trained on 16 kHz frequency, a speech model that accepts a float array corresponding to the raw waveform of the speech signal. Then, we will convert the audio to text, passing the prediction to the tokenizer decode to get the transcription. We are utilizing the LibriSpeech ASR Corpus, utilizing the clean testing data as the dataset that we are working with.

The LibriSpeech Dataset comes in nested folders which include a transcription file, and several audio files that correspond with different lines within the transcription file. We wrote a script to convert all of the .flac audio files into the .wav format which we will extract features from to pass into the model. The converted data was placed into the repository, and split into a smaller one alongside with the full dataset (for ease of testing) to focus on easier models to see the inner workings.

We then develop a function to predict the expected transcription by preprocessing the audio by running it through noise reduction scripts(to nullify any noises that would overwhelm the main audio that we will be wanting to focus i.e(traffic)) and then convert the file into a 16khz torchaudio object in order to tokenize and extract features. Using the configured model, we generate transcription predictions based on the embeddings of the audio file.

Any configuration of the Wav2Vec2 model can be evaluated by averaging the WER between the predicted transcription and the true transcription of an audio file for every audio file in the testing data. We can then compare different configurations of the model by comparing the average word error rate between configurations.

We utilized gridsearch to evaluate several key hyperparameters in order to test the difference in performance between their values. We evaluated the different models based on the Word Error Rate (WER) between the predicted sentences and the true transcription of the audio files. Upon running gridsearch across different hyperparameters, we sort a dataframe containing the results of different hyperparameter combinations in order to find the best performing set.

Upon performing research online, we decided that the model hyperparameters that we wanted to focus on tuning would be: attention_dropout, hidden_dropout, feat_proj_dropout, and mask_time_prob. We hypothesized that adjusting the dropout rate probabilities would change how much the model would "forget" as it iterated over the audio clip in order to take in new meaningful information, and potentially improve the model performance. We also hypothesized that adjusting the mask time probability would impact model performance, due to how it would change the percentage of feature vectors that would be masked, changing the information that the model would be working with which could lead to potential optimizations.We evaluated several trials where we would modify the different parameters and would let it run its course and see if the new parameters would change the WER that it would output.

Result & Analysis

	attention_dropout	hidden_droupout	feat_proj_dropout	mask_time_prob	WER
1	0.1	0.1	0.0	0.05	0.041336
58	0.5	0.1	0.2	0.05	0.041336
57	0.5	0.1	0.0	0.70	0.041336
56	0.5	0.1	0.0	0.30	0.041336
55	0.5	0.1	0.0	0.05	0.041336
...	...	...	...	...	...
24	0.1	0.5	0.2	0.70	0.041336
23	0.1	0.5	0.2	0.30	0.041336
22	0.1	0.5	0.2	0.05	0.041336
40	0.3	0.3	0.2	0.05	0.041336
81	0.5	0.5	0.4	0.70	0.041336

We found that there is no significant change in model performance with hyperparameter changes. We evaluated 162 different configurations of the Wav2Vec2 model which resulted in nearly identical results in terms of performance. The word error rate was 3.92% when testing on our smaller testing set, and was 4.13% when running on the full clean testing dataset from LibriSpeech (the same testing dataset that the original researchers of Wav2Vec2 used to evaluate their model).

As we stated before, the most significant hyperparameter for test performance is hidden_dropout and the reason for that was the usefulness of hidden_dropout. What hidden_dropout does is “The dropout probability for all fully connected layers in the embeddings, encoder, and pooler.” The main advantage of this method is that it prevents all neurons in a layer from synchronously optimizing their weights. This adaptation, made in random groups, prevents all the neurons from converging to the same goal, thus decorrelating the weights.

After the evaluation of 162 different configurations of the Wav2Vec2 model, there was no meaningful change in the performance. This is likely due to the fact that we were utilizing the Wav2Vec2-960h model, which was already fine-tuned on 960 hours of labeled English transcriptions from the LibriSpeech dataset, and the large amount of fine-tuning on the model was able to generate weights/parameters that were able to work extremely well regardless of the hyperparameters we adjusted. This was a surprising result to discover, because we were expecting modifying hyperparameters to affect model performance much more, but these findings truly highlight the power of the transformer based Wav2Vec2 model, especially when fine-tuned with significant amounts of data. These results also provide us with a strong jumping off point for future progress with optimizing audio to text models, specifically utilizing transfer learning on a less fine-tuned version of Wav2Vec2 in order to further optimize model performance.

Future direction

For future directions with this project, we want to further optimize the performance of the Wav2Vec2 model. We would do this by starting with a earlier checkpoint of Wav2Vec2 (either 10 minute or 100 hour fine-tuned models), and then perform downstream training by splitting our dataset into training, validation, and testing subsets, and fine-tune the model by adjusting the learning rate, vocabulary size, etc. in order to improve overall performance and reduce WER.

In addition to this, we would like to complete the integration between the front-end website and the model prediction function on the back end. This might entails that Flask would have to be install and imported into the frontend and backend to connect the two ends, and make it into a full-stack project. This would enable external users without access to the source code to utilize the power of Wav2Vec2 in order to convert their audio to text. This would streamline the user experience for people to access our research.

Resources

Wav2Vec2

LibriSpeech Dataset

Blog

Speech Recognition using Transformers in Python

Netflix Automated Subtitles

How to Build a Real-Time Transcription App in Python

Automatic Subtitle Synchronization through Machine Learning

How to Perform Real-Time Speech Recognition with Python

How to Create Subtitles for any Video with Python

Adding closed captions and subtitles

Automated Audio Captioning

Research Paper

NLP Driven Ensemble Based Automatic Subtitle Generation and Semantic Video Summarization Technique

Project Submission Details

GitHub Repository

Work Distribution

Macy So - In charge of building the front end for the platform, linking our model's capabilities online with Flask. Conducting background research, and the introductory abstract of the write-up.

Lazaro Solorzano - Responsible for developing the gridsearch algorithm for testing hyperparameters, performed extensive research for preprocessing (noise reduction), model choice, different hyperparameters to test, and datasets to use for training & testing.

Jared Chou - Responsible for organizing group meetings, initializing the pre-trained checkpoing of the Wav2Vec2 model. Organized and handled preprocessing of dataset, developing a model prediction function, as well as developing an evaluation function for the model based on average word error rate.