Goodnight Moon, Hello Early Literacy Screening

Solution write-up

1. Who are you (mini-bio) and what do you do professionally?

My name is Igor Ivanov. I'm a deep learning engineer from Dnipro, Ukraine. I specialize in CV and NLP and work at a small local startup.

2. What motivated you to compete in this challenge?

In the first place it was an intriguing multimodal dataset allowing to extract different features. Also speech competitions are relatively rare, so it was a nice opportunity to apply current state-of-the-art models like Whisper.

3. High level summary of your approach: what did you do and why?

My approach is based on the PyTorch and Huggingface Transformers. I used 4 modalities of the data each of which brings significant improvement for the model:

- original audio
- audio generated from expected text (by SpeechT5)
- original expected text
- text transcribed from original audio (by Whisper finetuned on positive examples)

To process these 4 components I built a multimodal classifier which uses the encoder from the finetuned Whisper model as audio feature extractor and Deberta-v3 as text feature extractor. The best single multimodal classifier is based on `microsoft/deberta-v3-base` plus `openai/whisper-medium.en` or `openai/whisper-medium`. My best final submission is an ensemble of 6 multimodal classifiers each of which uses `microsoft/deberta-v3-base` and different versions of Whisper, namely:

- `openai/whisper-medium.en`
- `openai/whisper-medium`
- `openai/whisper-small.en`
- `distil-whisper/distil-medium.en`
- `distil-whisper/distil-large-v3`
- `distil-whisper/distil-large-v2`

3.1 Original audio

I listened to several hundred training examples. The most complicated task by nature is nonword repetition. Also there are some very hard cases where the speech is unclear and

blurry. Eventually I decided to use raw audio without any cleaning or preprocessing because Whisper was pretrained on a very large and versatile dataset.

3.2 Audio generated from expected text (Text-to-Speech)

I generated speech from expected text using the SpeechT5 `microsoft/speecht5_tts` model with `microsoft/speecht5_hifigan` vocoder. As a speaker embedding I manually choose a vector from the `Matthijs/cmu-arctic-xvectors` dataset (index 7900). It is a female voice close to voices found in original audio. I tried to use different speaker embedding for each example including male and female voices with different properties, but the result was not better than the single speaker mentioned above.

3.3 Expected text

I cleaned expected text by applying the following processing: lowercase, remove outer spaces, remove all characters except letters and spaces.

3.4 Transcription (Speech-to-Text)

Given that positive labels in our dataset indicate that provided expected text matches the speech in the audio file, the natural idea is to finetune the transcription model on these examples. There are about 18k positive examples which is large enough. Finetuning gives us two advantages: we can obtain transcription of the audio and use it in addition to the expected text, and also finetuning improves the encoder part of the Whisper model, which we will use later in multimodal classifier.

I created a 5-fold cross-validation split to be able to predict the full training set using 5 out-of-fold parts. I used the `WhisperForConditionalGeneration` wrapper, `AdamW` optimizer and `CrossEntropyLoss` loss. I trained 4 epochs with a learning rate of `1e-5`. Best models were selected based on the loss, but I also computed WER and CER for evaluation purposes (see tables below).

3.5 Multimodal classifier

When all data components are ready, we process them with a multimodal classifier. I just concatenated original audio with generated audio, and original text with generated text separated by a single space. Multimodal classifier uses encoder from the finetuned Whisper model as audio feature extractor and Deberta-v3 as text feature extractor. To obtain audio features I extracted the last hidden state from the Whisper encoder and computed the average. To obtain text features I extracted the first token (CLS) from the last hidden state of Deberta. Extracted audio and text features are concatenated and processed by a fully connected fusion layer. Dimensions of audio features are from 768 to 1280 (depending on Whisper size) and dimension of text features is 768. As a fusion layer I chose a fully connected layer of size 256. I use ReLU as a fusion layer activation. The final layer has dimension 2 and outputs final logits.

I trained a multimodal classifier using the same exact 5-fold split which was used for transcription tuning to avoid data leakage. Only the first fold (index 0) is used in the final

solution. I trained each model using 7 sets of 2 epochs each with `Adam` optimizer and `1e-5` learning rate. I.e. I just rerun the same training command 7 times and selected the model with the best validation score. The reason for such an approach is that the model demonstrates fast convergence (only 2 epochs) with moderate variation in the validation scores. I tried lower learning rates and different schedulers (e.g. cosine annealing) but eventually the rerun-and-select strategy turned out to be better. When using this approach it is possible to develop a slight overfitting to the validation set, but in my experiments better local scores were still well correlated with leaderboard scores.

3.6 Ensemble

For the ensemble I used 6 models (mentioned at the beginning) and computed the average of predicted probabilities.

4. Do you have any useful charts, graphs, or visualizations from the process?

I concentrated on numerical optimization and used visuals only a couple times to look at Whisper spectrograms. So I don't have useful graphs.

5. Copy and paste the 3 most impactful parts of your code and explain what each does and how it helped your model.

1. The most impactful part of my code is the multimodal classifier. It extracts audio and text features and then processes them jointly to classify an example. I provide a definition of the forward pass.

```
def forward(self, images, input_ids, attention_mask):
    # Extract audio features
    image_features =
self.image_feature_extractor(images).last_hidden_state
    image_features = image_features.mean(dim=1)
    # Extract text features
    text_outputs = self.text_feature_extractor(input_ids=input_ids,
attention_mask=attention_mask)
    text_features = text_outputs.last_hidden_state[:, 0, :]
    # Fuse
    combined_features = torch.cat((image_features, text_features),
dim=1)
    combined_features = self.fc_fusion(combined_features)
    combined_features = self.relu(combined_features)
    # Cls
    output = self.classifier(combined_features)
    return output
```

2. Also I cleaned all expected text and generated transcriptions using the following code. Lowercase, trim outer spaces, leave only letters and spaces.

```
def clean(x):
    return re.sub('[^a-zA-Z ]+', '', x.strip().lower())
```

6. Please provide the machine specs and time you used to run your model.

Hardware:

- 12x CPU
- 32 GB RAM
- 1x RTX-3090-24GB GPU
- 500 GB SSD

Software

- Ubuntu 22.04
- Python: 3.10.12 (Conda)
- CUDA 12.4

Time:

- Training time: **140 hours**
- Inference time: **1 hour**

Note 1: It's not possible to train 2 of 6 models used in my solution on 16 GB GPU. Specifically, `distil-whisper/distil-large-v3` and `distil-whisper/distil-large-v2` cannot be trained on `T4-16GB` and `V100-16GB` even with batch size 1 and mixed precision.

Note 2: I successfully tested the solution on the following GPUs: `RTX-3090-24GB`, `L4-24GB`, `A100-40GB`.

Note 3: I provide bash scripts with a suffix `_a100` adapted for training on `A100-40GB` GPU which can speed up the training by about 2 times.

7. Anything we should watch out for or be aware of in using your model (e.g. code quirks, memory requirements, numerical stability issues, etc.)?

1. It's not possible to train 2 of the 6 models used in my solution on a 16 GB GPU. Please see section 6 above for details.

2. There is a well known warning from Huggingface tokenizers: `The current process just got forked …` related to possible excessive parallelism. I mitigate this warning by explicitly allowing parallelism: `os.environ['TOKENIZERS_PARALLELISM'] = 'true'`. I did not have any issues with this setting, but just in case of any problems it is possible to replace `true` with `false` at the beginning of the scripts to get more conservative behaviour.

3. For many complicated examples with unclear speech, especially for nonword repetition task, Whisper tends to hallucinate. For example if expected text is `koovnorb` the audio transcription may take the following forms:
   a) repetition of the expected text: `koovnorbkoovnorbkoovnorb`

b) random letters after the expected text:
`koovnorbteevohkmerforbuhkahydohfteevohk`
I did not apply any special means to mitigate hallucination because the Deberta model, which processes concatenated expected text and transcription, can easily identify useful information i.e. the beginning of the transcription.

4. Within my code in variable names and comments I use the terms `image` and `audio` interchangeably in relation to audio data. And in contrast I use the term `text` to denote textual information.

5. All relevant scripts accept parameters `batch_size` and `accum`, where `accum` means number of steps for gradient accumulation. Please note, that it is possible to adjust this parameters based on target hardware, but resulting batch size (i.e. product of `batch_size * accum`) should stay the same. Specifically, to finetune the transcription model (script `train_trans.py`) I used the resulting batch size 36 (e.g. `batch_size=4, accum=9`). To train a multimodal classifier (script `train_cls.py`) I used resulting batch size 32 (e.g. `batch_size=8, accum=4`).

8. Did you use any tools for data preparation or exploratory data analysis that aren't listed in your code submission?

No. I did not use any additional tools. At the exploratory data analysis stage I mostly listened to the provided audio to develop some cleaning or preprocessing methods. Eventually I decided to use original audio unchanged, because Whisper was trained with very large and versatile data.

9. How did you evaluate performance of the model other than the provided metric, if at all?

For local evaluation I used the same stratified 5-fold split for both transcription model training and multimodal classifier training and used out-of-fold predictions to compute metrics. For the transcription in addition to the cross-entropy loss I computed WER and CER metrics. For the multimodal classifier I calculated accuracy in addition to the log loss. Local log loss scores were very close (slightly better) and consistent with leaderboard scores. Accuracy closely followed changes in log loss. Please see `Table 1` and `Table 2` for all metric results.

10. What are some other things you tried that didn't necessarily make it into the final workflow (quick overview)?

My best multimodal classifier uses Whisper encoder as audio feature extractor and Deberta-v3 as text feature extractor. I tried the following modifications without improvement:

1. Add another audio feature extractor (`Wav2Vec2`)
2. Add image model (`EfficientNet-B7`) as audio feature extractor based on mel-spectrograms
3. Replace text feature extractor with: `mDeberta` (multilingual), `Bert`, `Roberta`, `XLM-Roberta` (multilingual), `SentenceTransformer`.

4. Given that Whispr accepts a fixed sampling rate of 16 kHz, I tried to resample audio into other sampling rates which effectively makes speech slower or faster: 8 kHz, 24 kHz, 32 hHz.
5. In the final submission I generated speech from expected text using a single female voice (i.e. the same speaker embedding), but I also tried to generate a different voice for each example without improvement.

11. If you were to continue working on this problem for the next year, what methods or techniques might you try in order to build on your work so far? Are there other fields or features you felt would have been very helpful to have?

To continue work on this problem first of all I would try more different feature extractors for audio. Also I would be interested in researching features extracted using signal processing techniques.

12. What simplifications could be made to run your solution faster without sacrificing significant accuracy?

Easy simplification of my solution is to use only the single best model (based on `whisper-medium.en` or `whisper-medium`) instead of the all 6 models. Given that the best model is not the largest, training time may be reduced by almost 10 times with very little log loss degradation around 0.25 vs 0.21. I provide adaptation of my solution for a single model via scripts with the `_single` suffix. Please see the last section of the notebook.