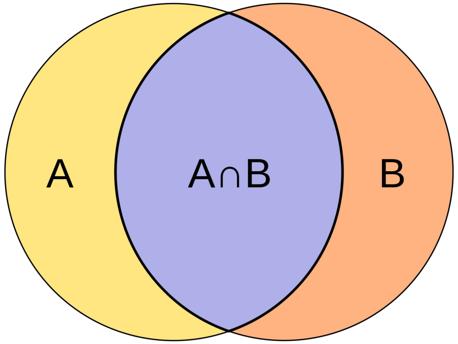
ID2223 - Lab 2

Group 27: Marie Gotthardt, Samuel Härner

Utilization for our Model: Recognition of Swedish Christmas Songs

- Created an app that automatically identifies Swedish Christmas songs based on the transcriptions generated by our model

- Method:
  - Shingle transcription and lyrics into sets of 5-grams and use Jaccard similarity as metric for document similarity
  - Jaccard similarity: size of intersection divided by size of union of sets
  - Use **MinHashing** for compressing shingle set and estimating the **Jaccard similarity** with the fraction of same entries in MinHash signatures



https://en.wikipedia.org/wiki/Jaccard_index#/media/File:Intersection_of_sets_A_and_B.svg

System Overview & Development Environment

- **System Components**
  - Feature Pipeline: prepare data for model (does not require GPU)
  - Training Pipeline: finetune model on data (requires GPU)
  - Inference Program: model creates transcription, minhash to match to lyrics

- **Development Environment**
  - Used Google Colab for running the pipelines and getting GPU access
  - Used Google Drive as the feature store
  - Used HuggingFace Space to run our inference program as an app

Experiments for Performance Optimization

- Create baseline model for comparison
    - lr = 1e-5, scheduler: linear, warmup steps = 200, dropout = 0.0, training steps = 1000
- Data-centric optimizations
    - Data augmentation tutorial: https://wandb.ai/parambharat/whisper_finetuning/reports/Fine-Tuning-Whisper-ASR-Models---VmlldzozMTEzNDE5
        - Gaussian noise
        - Time stretch
        - Pitch shift
- Model-centric optimizations
    - adding dropout (dropout = 0.1)
    - using different learning rates (learning rate = 1e-4, 1e-5)
    - using different learning rate schedulers (linear learning rate scheduler, cosine with hard restarts scheduler)
    - using different numbers of warmup steps (10, 20, 200)

Experimental Results

Baseline

Checkpoint	200	400	600	800	1000
WER	59.32	60.32	46.04	47.68	63.25

Data-centric Optimizations:

- Baseline training setup, but with augmented data: **best WER = 47.91 (500 steps), final WER = 65.20**

  => similar performance as baseline

- lr=1e-4, scheduler: linear, warmup steps = 10, dropout = 0.1, training steps = 1000 + augmented data: **best WER = 27.40 (800 steps), final WER = 31.09**

  => similar performance as corresponding experiment without augmented data

Experimental Results

Baseline

Checkpoint	200	400	600	800	1000
WER	59.32	60.32	46.04	47.68	63.25

Model-centric optimizations:

- Linear lr scheduler yielded satisfying results:
  - lr = 1e-4, warmup steps = 10, dropout = 0.0, training steps= 300: **WER = 23.95**
  - lr = 1e-5, warmup steps = 20, dropout = 0.1, training steps = 200: **WER = 25.53**
  - lr = 1e-4, warmup steps = 10, dropout = 0.1, training steps = 800: **WER = 27.87**

- Cosine with hard restarts lr scheduler did not work well for us
  - lr = 1e-4, warmup steps = 200, dropout = 0.0, training steps = 1000: **WER = 103.39**

Experimental Results: Baseline vs. Final Model

- lr = 1e-5, scheduler: linear, **warmup steps = 200, dropout = 0.0**, training steps = 1000

Checkpoint	200	400	600	800	1000
WER	59.32	60.32	46.04	47.68	63.25

- lr=1e-5, scheduler: linear, **warmup steps = 20, dropout = 0.1**, training steps = 1600

Checkpoint	200	400	600	800	1000	1200	1400	1600
WER	25.21	23.69	22.66	22.10	22.34	22.30	21.91	21.76

Discussion of our Results

- Considerable improvements from baseline model (best WER = 46.04) to final model (best WER = 21.76)
    - Final WER is acceptable for speech to text models [1]

- Better performance than the Whisper Small Swedish Fast (WER = 62.69) and the Whisper Tiny Swedish (WER = 44.19) but worse than the Whisper Medium Sv (WER = 10.71) [2]

- Other state-of-the-art Wav2vec models (trained on the Swedish common voice dataset and additional data) achieve a WER as low as 6.47 [3]

[1] https://learn.microsoft.com/en-us/azure/ai-services/speech-service/how-to-custom-speech-evaluate-data?pivots=speech-studio
[2] https://paperswithcode.com/sota/automatic-speech-recognition-on-mozilla-75
[3] https://paperswithcode.com/sota/speech-recognition-on-common-voice-swedish

Discussion of our Approach

- Limited GPU access
  - Regularly checkpointing model weights
  - Limitations on experimentations/ number of training steps

- Limited collection of Swedish Christmas songs
  - List of included songs can be found in app and in our Readme
  - If a user sings a song not included, our app is not able to identify it

- MinHashing method does not guarantee good guesses
  - Users can increase likelihood of a correct guess by singing more clearly/ singing more of the song lyrics

Potential Improvements

- Finetune bigger model (e.g. Whisper-Medium, -Large)
- Use another model (e.g. a Wav2vec model)
- More data than just Common Voice dataset
  - Many high performing models also use the *NST Swedish ASR Database*: https://www.nb.no/sprakbanken/en/resource-catalogue/oai-nb-no-sbr-56/
- More warmup steps and longer training run (if GPU access not an issue)
  - Low warmup steps allowed us to achieve decent performance quickly (WER=25 at step 200)
  - High warmup steps might help reach a better local minimum

Dataset

- Dataset: mozilla-foundation/common_voice_11_0: Swedish
    - ~16,74 GB
    - 12360 training instances, 5069 test instances
    -

Feature Pipeline

- WhisperFeatureExtractor:
  - Pads/ truncates audio inputs to 30s snippets
  - Converts audio inputs to log-Mel spectrogram → required input format for the Whisper model
- WhisperTokenizer
  - Maps token ids (= output from Whisper model) to corresponding test string
  - Use pre-trained tokenizer
- WhisperProcessor
  - Wraps feature extractor and tokenizer
- Downsampling the sample rate
  - Downsample audio from 48kHz to 16kHz (required sample rate for the Whisper model)

Training Pipeline

- Data Collator
  - Input features are handled by the FeatureExtractor, labels are handled by the Tokenizer
  - Returns batched PyTorch tensors
- Evaluation Metrics
  - Word Error Rate (WER)
- Load pre-trained Whisper small checkpoint
- Define training configuration & trainer
- Train the model