Speech-to-text models to transcribe emergency calls

Øyvind Grutle & Jens A. Thuestad

KoKom

HELSE BERGEN
Haukeland universitetssjukehus

Western Norway
University of
Applied Sciences

UNIVERSITAS BERGENSIS

MMIV

Forskningsrådet

TROND
MOHN
STIFTELSE

Background

AI-Support in Medical Emergency Calls

- Emergency Medical Communication Centres (EMCC)
- Stroke patients
    - 60% accuracy

"Time is brain"

- Camilo R. Gomez

- Our part: Transcribe emergency calls

Machine learning for speech recognition

- Automatic speech recognition (ASR)
  - Since 1952 - the Audrey system
  - Neural networks

- Speech-to-text
  - Siri, YouTube, Google Assistant, etc
  - Norwegian solutions still lacking

Challenge

Accurate transcriptions of emergency calls

- Norwegian speech
- Unclear audio
- Available data

Multilingual Whisper models

Approach

- Whisper
- Hugging Face
  - Inference API
  - Transformer library
- Web-app

Training the model

```
from transformers import WhisperForConditionalGeneration, WhisperFeatureExtractor, WhisperTokenizer, WhisperProcessor, Seq2SeqTrainingArguments, Seq2SeqTrainer

model = WhisperForConditionalGeneration.from_pretrained( "openai/whisper-large-v2" )

feature_extractor = WhisperFeatureExtractor.from_pretrained( "openai/whisper-large-v2" )

tokenizer = WhisperTokenizer.from_pretrained( "openai/whisper-large-v2" , language="norwegian", task="transcribe")

processor = WhisperProcessor.from_pretrained( "openai/whisper-large-v2" , language="norwegian", task="transcribe")

training_args = Seq2SeqTrainingArguments(
   output_dir=REPO_NAME,  # change to a repo name of your choice
   per_device_train_batch_size =16,
   gradient_accumulation_steps =1,  # increase by 2x for every 2x decrease in batch size
   learning_rate=5e-6,
   warmup_steps=2,
   #max_steps=100,
   num_train_epochs=10,
   gradient_checkpointing =True,
   fp16=True,
   evaluation_strategy ="epoch",
   save_strategy="epoch",
   per_device_eval_batch_size =8,
   predict_with_generate =True,
   generation_max_length =225,
   #save_steps=20,
   #eval_steps=20,
   logging_steps=1,
   report_to=["tensorboard"],
   load_best_model_at_end =True,
   metric_for_best_model ="wer",
   greater_is_better =False,
   push_to_hub=True,
)

trainer = Seq2SeqTrainer(
   args=training_args,
   model=model,
   train_dataset=dataset["train"],
   eval_dataset=dataset["test"],
   data_collator=data_collator,
   compute_metrics=compute_metrics,
   tokenizer=processor.feature_extractor,
)

trainer.train()
```

Training the model

```
from transformers import WhisperForConditionalGeneration, WhisperFeatureExtractor,
WhisperTokenizer, WhisperProcessor, Seq2SeqTrainingArguments, Seq2SeqTrainer

model = WhisperForConditionalGeneration.from_pretrained("openai/whisper-large-v2")

feature_extractor = WhisperFeatureExtractor.from_pretrained("openai/whisper-large-v2")

tokenizer = WhisperTokenizer.from_pretrained("openai/whisper-large-v2",
language="norwegian", task="transcribe")

processor = WhisperProcessor.from_pretrained("openai/whisper-large-v2",
language="norwegian", task="transcribe")
```

Training the model

```
training_args  = Seq2SeqTrainingArguments(
   output_dir =REPO_NAME ,  # change to a repo name of your choice
   per_device_train_batch_size  =16,
   gradient_accumulation_steps  =1,  # increase by 2x for every 2x decrease in batch size
   learning_rate =5e-6,
   warmup_steps =2,
   #max_steps=100,
   num_train_epochs =10,
   gradient_checkpointing =True,
   fp16=True,
   evaluation_strategy ="epoch" ,
   save_strategy ="epoch" ,
   per_device_eval_batch_size  =8,
   predict_with_generate =True,
   generation_max_length =225,
   #save_steps=20,
   #eval_steps=20,
   logging_steps =1,
   report_to =["tensorboard" ],
   load_best_model_at_end =True,
   metric_for_best_model ="wer",
   greater_is_better =False,
   push_to_hub =True,
)
```

Training the model

```
trainer = Seq2SeqTrainer(
   args=training_args,
   model=model,
   train_dataset=dataset["train"],
   eval_dataset=dataset["test"],
   data_collator=data_collator,
   compute_metrics=compute_metrics,
   tokenizer=processor.feature_extractor,
)

trainer.train()
```


AMK SPEECH-TO-TEXT

Loading the models

```
whisper_pipeline = pipeline(
 "automatic-speech-recognition",
 model="amk-whisper-local",
 chunk_length_s=30,
 device=device,
 generate_kwargs={
       "language":"<|no|>",
       "task":"transcribe"
 }
)

dz_pipeline = Pipeline.from_pretrained(
   'pyannote/speaker-diarization',
   use_auth_token="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"
   )
```

Handling the requests

```
@app.route('/transcribe', methods=['POST'])
def transcribe_audio():
    audio_file = request.files['file']
    file_name = "temp-data/"+audio_file.filename
    audio_file.save(file_name)

    transcription = whisper_pipeline(file_name, return_timestamps=True)["chunks"]
    dz = dz_pipeline(file_name, min_speakers=2, max_speakers=5)

    os.remove(file_name)

    speaker_list = get_speaker_list(dz)

    labeled_transcriptions = label_transcriptions(transcription, speaker_list)

    srt_text = generate_srt_text(labeled_transcriptions)
    doc_text = generate_doc_text(labeled_transcriptions)

    data = {}
    data['doc_text'] = doc_text
    data['srt_text'] = srt_text


    return jsonify(data)
```

Audio transcription

```
const transcribeAudio = async (file: Blob) => {
       setLoading(true);
       setDocTranscript("");
       if (!file) {
           setLoading(false);
           return;
       }
       try {
           const response = await API.transcribeAudio(file, audioFile!.name);
           setLoading(false);
           setDocTranscript(response.doc_text);
           setSrt(response.srt_text);
       } catch (error) {
           toast({
               title: "An error has occured." ,
               description: "An error occurred during the transcription. Try again later." ,
               status: "error",
               duration: 5000,
               isClosable: true,
             });
       }
   };
```

Results

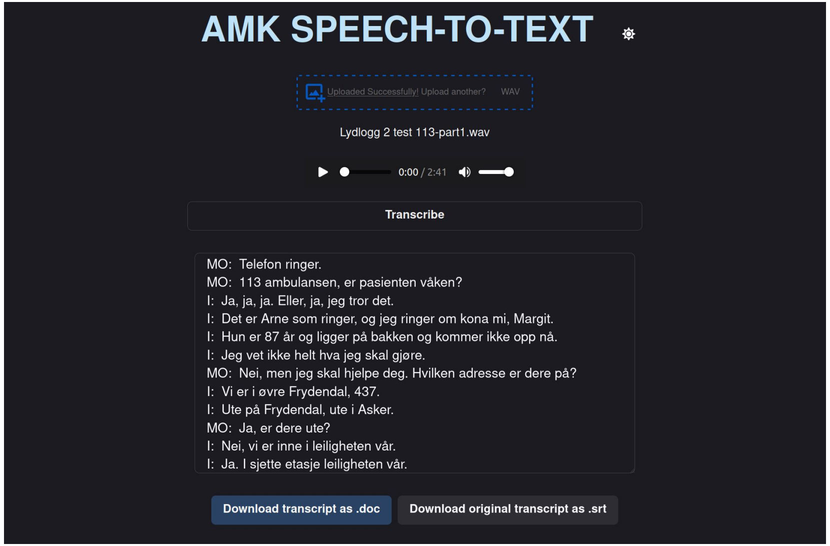
- Fine-tuned model
- Web App

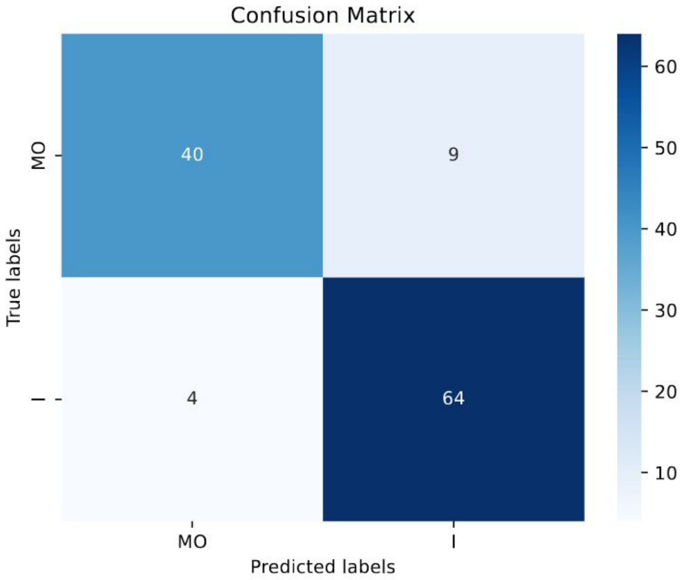
Model	WER
Fine-tuned	32.9443 %
Whisper large-v2	33.4829 %

```
Spoken:
"My name is Paul and I
am an engineer"

Model 1 prediction:
"My name is ball and I
am an engineer"
WER: 11.11%

Model 2 prediction:
"My name is Paul and
I'm an engineer"
WER: 22.22%
```

Ground truth	Fine-tuned	Whisper
MO: Ja. Er det sånn at hun føler hun holder på å besvime?	MO: Ja, er det sånn at hun selv holder på å besvime?	MO: Er det sånn at hun selv holder på å befime?

Speaker Tagging

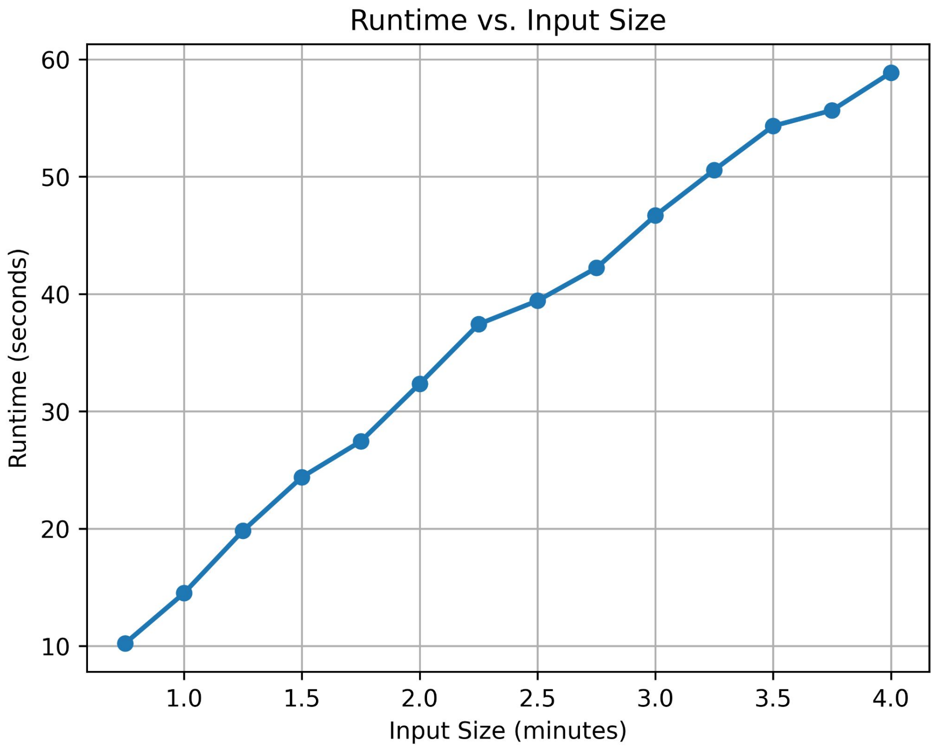




Transcription Experiment

	Semi-automatic	Manual	Improvement
Author 1	23:07 (Part 2)	31:17 (Part 1)	8:10 (35.33 %)
Author 2	15:49 (Part 1)	24:43 (Part 2)	8:54 (56.26 %)

Transcription Time



Limitations

- Trained on simulated data
- Sensitive data
- Required hardware

Thank you!

KoKom

HELSE BERGEN
Haukeland universitetssjukehus

Western Norway
University of
Applied Sciences

UNIVERSITAS BERGENSIS

MMIV

Forskningsrådet

TROND
MOHN
STIFTELSE