1 Data Preparation

? Do you see anything missing in the text transcription? How would it affect our end goal of performing Speech Translation?

The transcript:

> CHAPTER ONE MISSUS RACHEL LYNDE IS SURPRISED MISSUS RACHEL LYNDE LIVED JUST WHERE THE AVONLEA MAIN ROAD DIPPED DOWN INTO A LITTLE HOLLOW FRINGED WITH ALDERS AND LADIES EARDROPS AND TRAVERSED BY A BROOK

Comparing the audio sample, and its corresponding transcript reveals, misrepresentation of vocal words. For example, the word *Mrs.* is misspelled as *MISSUS*. A non-standard spelling like *MISSUS* might not be as well-represented in the training data, which could lead to less accurate recognition if the system is not robust to such variations. Moreover, ASR systems use language models to predict the likelihood of sequences of words. If "MISSUS" is not common in the language model's training data, the system might be less likely to recognize and transcribe it correctly.

? Can you explain the shape you see?

Figure 2 shows a spectrogram. The spectrogram depicts frequency on the vertical axis and time on the horizontal axis. A higher amplitude at a specific time and frequency is represented by a brighter color in the spectrogram. Time intervals (0-50, 350-500) without bright colors signify pauses in speech. Conversely, intervals with brighter colors indicate active speech. The content of the speech can be inferred based on the amplitudes of frequencies over time. Figure 1 illustrates an example where the spoken content is inferred.

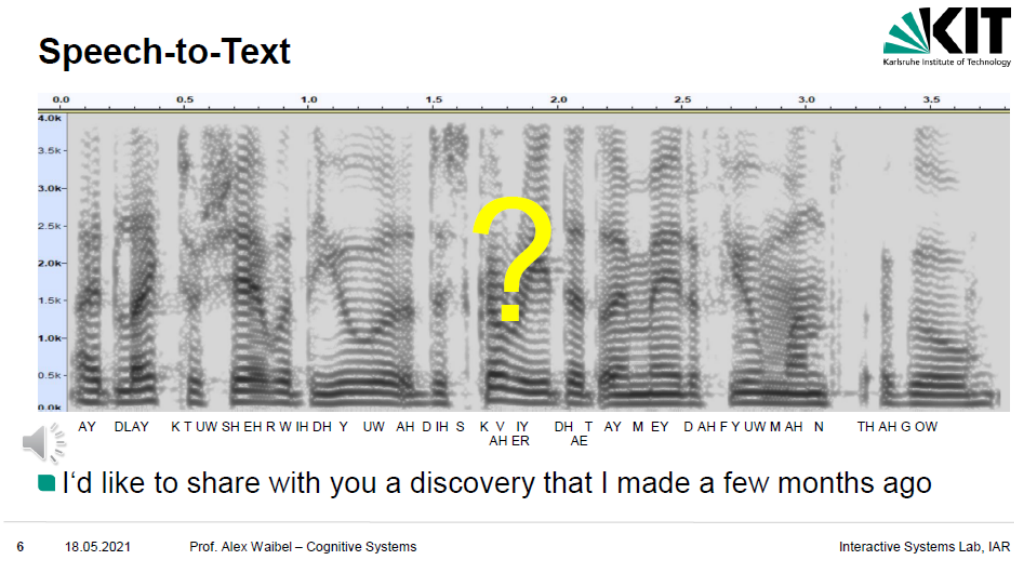

Figure 1: Spectrogram with inference of the spoken language. Waibel [2021]

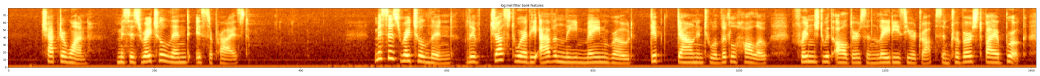

Figure 2: Log Mel Filter Bank Filters

2 Training

? Why do we need a much larger max-tokens than when training a Machine Translation model?

In the context of training a Speech-to-Text (STT) Transformer, there's a specific reason why a higher max-tokens value is required compared to training a Machine Translation model. This difference stems from the distinct characteristics of the input data in STT models.

In STT models, the primary input is audio waveforms. These waveforms, by their nature, are considerably more extended than the typical sequences of tokens you'd find in machine translation tasks. This is because the process of tokenizing for STT involves converting audio – which comes in extended, continuous forms – into a series of tokens representing the spoken language. This tokenization translates the audio into smaller, manageable units, subsequently converting these into tokens for the model to interpret and learn from.

Given the lengthiness of audio data, the token sequences created for STT models are inherently longer than those in machine translation. Hence, a larger max-tokens value is essential. A smaller max-tokens might lead to the truncation of these sequences, potentially resulting in the loss of critical information and negatively impacting the model's efficacy. Thus, the increased max-tokens value in STT training is a critical factor for accommodating the extended length of token sequences derived from audio inputs, ensuring comprehensive processing without loss of vital data.

? Can you again tell from the training log: the number of encoder layers, the number of decoder layers, embedding size, number of trainable parameters...?

Encoder:
Number of encoder layers: 12 x *TransformeEncoderLayers*
Embeddings size: 256

Decoder:
Number of decoder layers: 6 x *TransformerDecoderLayerBase*
Embeddings size: 256

Output feature size: 900
In total, there are 27,206,656 shared parameters in the model.

? What are the difference comparing to the Machine Translation model that we have seen in the previous lab? Can you tell why?

Architecture: The Machine Translation (MT) model from the previous task consists of a *TransformerEncoderBase* for the encoder and a *TransformerDecoderBase* for the decoder and it uses Multihead Attention and Linear layers for various operations.
The current model is a modified version of the Transformer architecture. It uses an *S2TTransformerEncoder* for the encoder and a *TransformerDecoderScriptable* for the decoder. It also employs *Conv1dSubsampler* for subsampling and *Conv1d* layers in the encoder.
In general the Automatic Speech Recognition (ASR) model has a more complex encoder with 12 layers, while the machine translation model has 6 layers for both encoder and decoder.
ASR tasks benefit from a deeper encoder to capture intricate features in speech signals.

Embeddings: In the MT model, both encoder and decoder use an embedding layer with an embedding dimension of 512. In the ASR model, the encoder uses *Conv1dSubsampler* for subsampling, and the embedding dimension is 256. Both models use *SinusoidalPositionalEmbedding* for positional embeddings.

Layer Normalization:
Both models use *LayerNorm* for normalization in various parts of the architecture.

Dropout: Both models utilize dropout for regularization in multiple locations

Output Projection: In the MT model, the output projection is a Linear layer with an output dimension of 7600. In the ASR model, the output projection is a Linear layer with an output dimension of 900.

Number of Layers: The MT model consists of 6 layers in both the encoder and decoder, whereas the ASR model, with 12 layers in the encoder and 6 layers in the decoder, has double the amount of encoder layers.

Subsampling: The second model incorporates subsampling using *Conv1d* layers in the encoder.

Embedding Tokens: The MT model embeds 6104 tokens, while the second model embeds 900 tokens.

Kernel Sizes: The second model uses kernel sizes of 5 in its *Conv1d* layers for subsampling.

2.1 Training Dataset

As training dataset the LIBRISPEECH train-clean-100 dataset was used.

Corpus: The train-clean-100 subset specifically refers to a "clean" subset of training data, indicating that the audio recordings in this subset have undergone some form of pre-processing to remove noise or artifacts.

Size of Train-clean-100 (6.3GB): The "100" in train-clean-100 (100 hours clean speech) signifies a subset with a lower level of background noise compared to other subsets like train-other-500 (500 hours speech), which may include more diverse and noisy recordings.

Content: The dataset covers a diverse range of speakers, reading various types of texts, reflecting the diversity found in audiobooks.

The dataset is freely available for research purposes and can be accessed from the official LIBRISPEECH website or relevant repositories.

3 Evaluation

3.1 Dev Dataset

As dev dataset the LibriSpeech dev-clean dataset was used. The "dev-clean" subset of LibriSpeech is specifically a development set characterized by "clean" speech.

The purpose of the "dev-clean" subset is to provide high-quality, clear audio samples that are appropriate for the development and tuning of our ASR model.

The size of the dev-clean dataset is 337MB.

3.2 Test Dataset

The used test dataset is the LibriSpeech test-clean dataset. Also this dataset is a dataset characterized by "clean" speech.

The test-clean dataset has a size of 346MB.

? Is WER suitable for languages such as Chinese or Japanese?

Word Error Rate (WER) can be applied to languages like Chinese and Japanese, but it's important to note that the effectiveness of WER as a metric may vary depending on the specific characteristics of the language. Chinese and Japanese, for example, are character-based languages, and the segmentation of words is not as straightforward as in space-separated languages like English.

In Chinese, for instance, words are not separated by spaces, and characters represent meaningful units. Therefore, evaluating word-level error rates may not be as meaningful as it is in languages where words are clearly delimited by spaces.

In summary, while WER can be applied to Chinese and Japanese, it's essential to consider language-specific characteristics and, if necessary, explore alternative metrics such as CER or SER for a more accurate evaluation of ASR system performance.

We have faced difficulties regarding the printing the logs in the BW cluster in a file. We have tried many times with different approaches, and every time the job has failed to write us the logs files. On every attempt, we have received an empty log file. Having no access to our log files while in BW Cluster

Type	Beam Size	MEL WER Score	Wav2vec WER Score
Test Set	5	21.15	149.43

We were quite surprised by how poorly our training model performed when using wav2vec data. We spent a considerable amount of time setting up the environment in the BW Cluster and getting the model up and running. Unfortunately, the results we obtained from the wav2vec model were not at all what we had hoped for.

Noticing a significant difference between the Mel Spectrogram and the wav2vec representation. We have adapted the Mel Spectrogram model from using 80 features to utilize 768 features as input, so that we can use the same architecture but with wav2vec representations as input. This naive attempt to simply change the expected input size of the model however didn't yield the results we wanted.

Furthermore, we noticed a warning about uninitialized weights in the model, which we initially brushed aside. It's possible that this warning had something to do with the disappointing results we experienced.

Looking ahead, we're determined to address these issues and make advanced improvements to our model to achieve the performance we're aiming for. Any guidance in this regard would be highly appreciated.

4 Further Considerations

? Given the two models you've had by now (ASR and MT), what can you do next to have a working ST system?

For having a working ST system, next a cascaded speech translation could be applied. A cascaded ST system involve the following steps.

Step 1 - Speech Recognition The first stage involves converting spoken language into written text. This is known as speech recognition, and it's the process of transcribing spoken words into a textual representation.

Step 2 - Machine Translation The transcribed text is then translated from the source language to the target language using machine translation techniques.

? Is there any drawbacks with this approach?

Common drawbacks for cascaded speech translation systems are:

Error Propagation: Errors made in one stage of the cascade can propagate to subsequent stages, leading to cumulative inaccuracies. For example, if the speech recognition stage misinterprets a spoken word, the translation and synthesis stages will be based on this error.

Latency: Cascaded systems may introduce additional latency as each stage requires processing time.

Complexity: Cascaded systems tend to be more complex to design, implement, and maintain compared to end-to-end systems.

Resource Intensive: Training and running multiple models for different stages of the translation process can be resource-intensive, requiring more computational power and storage compared to end-to-end systems.

References

Prof Alex Waibel. Cognitive systems slides, 2021.