Karlsruhe Institute of Technology

Master Thesis: Predictive Speech Recognition and End-of-Utterance Detection Towards Spoken Dialog Systems

Master's Thesis of

Oswald Zink

Interactive Systems Lab (ISL)
Institut for Anthropomatics and Robotics (IAR)
KIT Department of Informatics

Reviewer:	Prof. Dr. Alexander Waibel
Second reviewer:	Prof. Dr. Tetsunori Kobayashi
Advisor:	M.Sc. Carlos Mullov
Second advisor:	Dr. Yosuke Higuchi

20. March 2024 – 20. Oct 2024

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

PLACE, DATE

. . . . . .

(Oswald Zink)

Acknowledgments

I would like to thank Prof. Dr. Alexander Waibel for the opportunity of writing my thesis at Waseda University in Japan. I would also like to thank my advisor Carlos Mullov for his quality research supervision and his profound technical understanding and Christian Huber for helping with technical problems. Furthermore, much heartfelt gratitude for the warm reception and great supervision from the Japanese side. I would like to thank Mr. Tetsunori Kobayashi for his academic guidance, hospitality and optimism in my research. Thanks to my advisor Yosuke Higuchi for his supervision, patience, hospitality and enriching discussions. Without the support of these people, this research would not have been possible.

This work was supported by the Baden-Württemberg-STIPENDIUM.

Abstract

Effective spoken dialog systems should facilitate natural interactions with quick and rhythmic timing, mirroring human communication patterns. To reduce response times, previous efforts have focused on minimizing the latency in automatic speech recognition (ASR) to optimize system efficiency.

However, this approach still requires waiting for the ASR module to complete processing until a speaker has finished speaking, which limits the time available for natural language processing (NLP) to formulate accurate responses. As humans, we continuously anticipate and prepare responses even while the other party is still speaking. This allows us to respond appropriately without missing the optimal time to speak.

In this work, as a pioneering study toward a conversational system that simulates such human anticipatory behavior, we aim to develop a model that can predict the forthcoming words and estimate the time remaining until the end of an utterance (EOU), using an incomplete utterance. To achieve this, we propose a training strategy for an encoder-decoder-based ASR system, which involves masking future segments of an utterance and prompting the decoder to predict the words in the masked audio. We use this model's inherent cross-attention to directly predict the EOU. To this end, we propose a cross-attention-based algorithm.

The experimental results demonstrate the proposed model's ability to predict upcoming words and estimate future EOU events up to 300ms prior to the actual EOU. Moreover, the proposed training strategy exhibits general improvements in ASR performance.

Zusammenfassung

Effektive Dialogsysteme sollten in der Lage sein, schnelle, rhythmische Antwortzeitpunkte, wie sie in der menschlicher Kommunikation üblich sind, zu finden. Um die Latenzen der Antworten zu reduzieren, lag der bisherige Forschungsschwerpunkt darauf, die Effizienz des automatischen Spracherkennungssystems (ASR) zu verbessern. Dieser Ansatz erfordert jedoch das komplette Transkript des ASR-Moduls, also erst, bis der Nutzer seine Aussage vollständig geäußert hat. Dadurch ist das nachfolgende Modul zeitlich limitiert eine adäquate Antwort zu generieren.

Menschen hingegen antizipieren und bereiten kontiniurlich eine Antwort vor, selbst wenn die andere Partie noch spricht. Dieses Verhalten erlaubt es Menschen zu einem passenden Zeitpunkt zu antworten.

In dieser Arbeit wollen wir Pionierarbeit für ein Konversationssystem leisten, welches als Grundlage solches antizipatorisches Verhalten hat. Unser Ziel ist es ein Modell zu entwicklen, das in der Lage ist mit unvollständigen Audiosignal, nachfolgende Worte zu vorherzusagen und die verbleibende Zeit bis zum Ende der Aussage (EOU) des Nutzers vorherzusagen.

Im Rahmen dessen, nutzen wir ein auf der Encoder-Decoder Architektur basierendes Modell und schlagen eine Trainingsmethode vor, welche Segmente des Audioinputs maskiert, sodass das Modell gezwungen ist zukünftige Wörter des maskierten Segments vorherzusagen. Zusätzlich, führen wir einen Algorithmus ein, der auf Cross-Attention beruht, um das EOU sowohl mit linguistischer als auch mit akustischer Information vorhersagen.

Die Experimente zeigen, dass unser vorgeschlagenes Model in der Lage ist zukünftige Worte, und das EOU innerhalb von 300ms bevor das tatsächliche EOU zu predizieren. Außerdem scheint die vorgeschlagene Trainingsmethode die ASR-Leistung generell zu verbessern.

Contents

List of Figures

List of Tables

1. Introduction

In natural human conversations, individuals frequently begin speaking immediately after or even before the other party finishes. This rhythmic and prompt flow of dialog is facilitated by their speculative ability to predict what the other will say next and anticipate when they will stop speaking, thus allowing for the formulation of timely and appropriate responses.

In contrast, current spoken dialog systems typically lack this capability, as they only begin to prepare responses after their automatic speech recognition (ASR) module has completed its transcriptions. This delay forces subsequent Natural Language Processing (NLP) modules — language understanding, dialog policy, and natural language generation — to produce responses within a limited timeframe, potentially compromising the quality and speed of interactions with users.

To address the delay inherent in ASR, recent research has focused on minimizing the latency of ASR systems, developing online streaming architectures [13, 15, 47, 23, 37, 50] that effectively control the number of look-ahead frames [25, 21, 24, 49, 44, 36]. Nonetheless, these models typically operate in real-time, requiring input audio information up to the final EOU frame. As a result, the overall latency of a dialog system experienced by users is inevitably above zero seconds. Simultaneously, the downstream NLP modules face challenges in achieving quick responses within the desired latency window, which can span from as early as 200 ms to 400 ms in natural human interactions [20, 8, 30].

Inspired by the speculative capabilities of humans, there have been efforts to anticipate future information in incomplete utterances, extending beyond the constraints of conventional real-time ASR systems. A predominant approach involves using an external language model to generate forthcoming words from a partial ASR hypothesis, as discussed in studies [8, 30, 46]. In [8, 30], the syntactic completeness of spoken text at mid-utterance is evaluated to represent the likelihood that the speaker will continue speaking, which has proven crucial for efficiently predicting turn-shifts and timing responses. Similarly, [46] has fine-tuned a large language model to generate future words from incomplete ASR outputs, effectively incorporating audio information from an ASR model through a soft prompting technique.

This work also aims to advance ASR systems with predictive functionality, aiming to provide the downstream NLP modules with ample time to operate and enable the dialog system to respond effectively. Uniquely, our approach functions as a straightforward extension of traditional ASR models, which can naturally incorporate both acoustic and linguistic contexts to enable future word and EOU prediction. Our work differs from [8, 30] as their work uses an external LM to predict EOU, thus not involving acoustic information. While [46]s work does include acoustic information, they rely on an external language model. We try to provide a solution that combines EOU prediction and future word prediction, without relying on external language models.

To this end, we propose an encoder-decoder-based ASR model [5, 2, 43] that is specifically constructed to simulate human predictive capabilities within a unified framework. We evaluate our model on EOU prediction and predictive ASR tasks.

Predictive EOU detection forecasts the future endpoint of an utterance. Predictive ASR, on the other hand, generates the complete transcription before the utterance concludes, thereby allowing the downstream NLP modules to start processing earlier for a low-latency response [3]. More concretely, for predictive EOU prediction, our approach leverages alignment information obtained from the cross-attention mechanism. We feed mid-utterance input followed by empty input into the model, and by analyzing the attention weights applied to the empty input, we predict future endpoints.

For predictive ASR, we use the decoder to produce tokens corresponding to the empty input, where the decoder functions as a generative language model to predict future tokens. In order for the model to operate even in the absence of future speech input, we design a training strategy that randomly masks segments of future speech, thereby facilitating the training of the language model capability in the decoder.

1.1. Problem statement

Research question 1: How well and how far into the future can we perform predictive EOU detection using the inherent cross-attention of encoder-decoder models? Since the ultimate goal is to reduce the latency of a dialogue system, we aim to research how well we can utilize the inherent cross-attention in encoder-decoder models to do EOU prediction, with a model trained with our proposed masked training.

Research question 2: How well and how far into the future can we predict future words? Analogous to the first research question, this thesis also investigates the ability to predict future words. Specifically, we aim to determine how far into the future predictions can be made and the quality of these predictions. Furthermore, we want to see if our proposed training method aids this task.

2. Fundamentals

This chapter covers all the relevant subjects for this thesis. Fundamental knowledge about Neural Networks are assumed and therefore omitted. We first discuss the attention mechanism in section 2.1. Afterwards, we look at the Transformer architecture, the Conformer variant of it in section 2.2 and their use of positional encoding in section 2.3. Furthermore, in section 2.4, we take a look at the hybrid CTC-attention based encoder-decoder model used by ESPnet, which we use for all of our experiments.

Finally, in section 2.5 we take a look at the bigger picture and discuss usual components and architecture choices of dialogue systems. This includes a discussion of streaming architectures, which are commonly employed in dialogue systems. The chapter concludes by introducing and analyzing the two key tasks of this thesis — future word prediction and end-of-utterance (EOU) prediction — in subsection 2.5.2 and subsection 2.5.3, respectively.

2.1. Attention

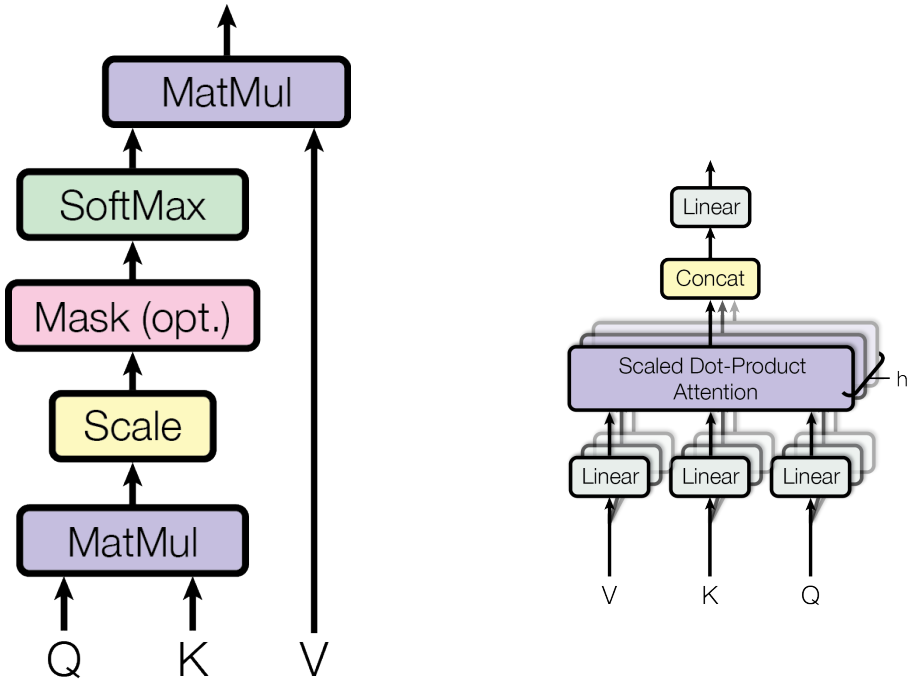
The Attention mechanism was first introduced by [1]. Later, the authors of [39] introduce a variation — the scaled dot product Attention — as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{2.1}$$

In the context of Attention, queries Q, key K and values V are all feature vectors with some dimensionality d_k for Q and K and d_v for V. Usually, $d_v = d_k$. Generally speaking, the queries Q represent the current features of interest and are matched against keys K for relevant information. The scaled dot product, computed by $\frac{QKT}{\sqrt{d_k}}$, computes the attention weights, which quantify the degree of matching, shown in Equation 2.1. These weights are then normalized using the softmax function to produce a probability distribution. The scaling $\frac{1}{\sqrt{d_k}}$ is added to stabilize gradients. The attention mechanism is visually shown in Figure 2.1a.

The authors of [39] have found it beneficial to use multiple smaller projections — Multi Head Attention (MHA) — instead of one single big projection, which remains state of the art. A visualization is shown in Figure 2.1b

Intuitively, the result of an Attention layer is a vector of information that is adjusted to the current situation, i.e. query q, potentially containing all necessary context information. Concretely, the softmax Attention scores α_{ij} are used to build a weighted sum c_i of values as shown in Equation 2.2.



(a) The attention mechanism visualized [39].

(b) Multihead-attention visualized [39].

Figure 2.1.: The attention and multihead attention mechanism visualized.

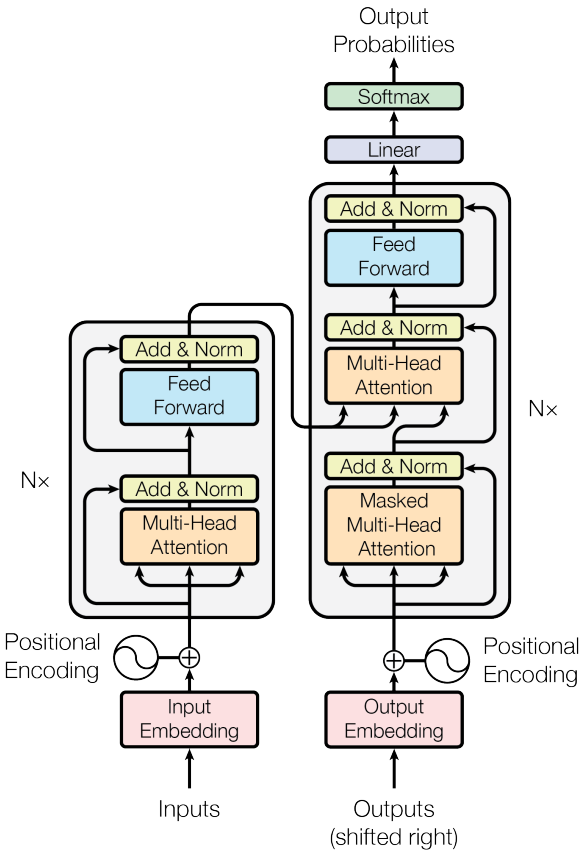
$$c_i = \sum_{j=1}^{K} \alpha_{ij} v_j \tag{2.2}$$

This Attention mechanism can be used for many different architectures. Most commonly, the attention mechanism is used as either self- or cross-attention. Essentially, the attention mechanism stays the same, just the queries Q the key K to match against differ. We will refer to this as "attending to somewhere". Cross-attention is used in encoder-decoder models and is discussed in further detail in section 2.4. Self-attention attends to the model itself, e.g., queries Q of an encoder attend keys K of the same encoder.

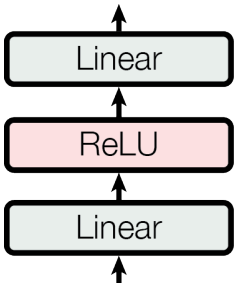
2.2. Transformer and Conformer

The Attention mechanism finds heavy use in transformers [39].

Transformer The Transformer, shown in Figure 2.2a is a neural network architecture that makes heavy use of the Attention mechanism. As shown in Figure 2.2 each block of the Transformer consists of a MHA layer and a two layer feed forward network, both of which are normalized and have a residual pathway added. The norm can also be applied before, instead of after, which tends to lead to more stable training. Furthermore, if used as a decoder, an additional MHA layer is used for Cross-Attention. Usually, multiple of these blocks are stacked on top of each other.



(a) The attention mechanism visualized [39].



(b) Feed forward network used in the vanilla Transformer [39].

Figure 2.2.: The "vanilla" transformer architecture and the FFN used in the transformer.

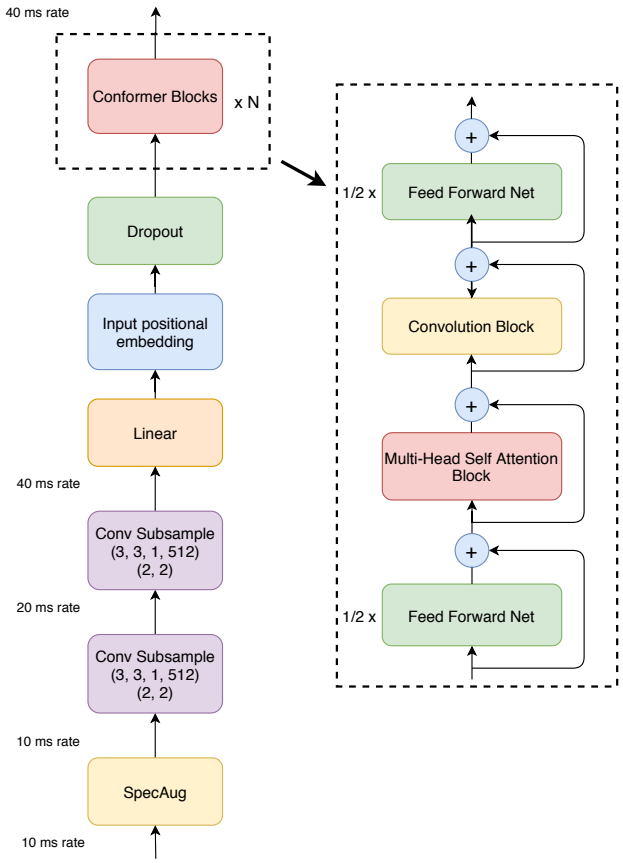


Figure 2.3.: The Conformer architecture [14]

Since the Transformer otherwise does not have any information about position, a positional encoding is added at the very first layer. Many different positional encodings exist, learned and fixed [10]. They are covered in section 2.3.

Finally, when used as decoder a last linear & softmax layer is added for the final output tokens.

Conformer A modification to the Transformer is the Conformer[14]. The architecture is shown in Figure 2.3. The architecture essentially tries to capture the best of the worlds of a Convolutional Neural Network (CNN) [40, 19], and a Transformer by modifying the Transformer Block to a Conformer block. As seen in Figure 2.3, a Conformer Block uses macaron-like FFN layers, i.e., instead of placing a two layer FFN at the end of the transformer block as in the "vanilla" transformer, they sandwich a Multi-Head-Self-Attention and a Convolutional block in between two FFN layers. Convolutional layers capture local information well, whilst MHA layers capture global information well. In ASR specifically, local information is used during processing and the Conformer has been outperforming the traditional Transformer [14].

Furthermore, the Conformer uses convolutional subsampling, which results in a change of the frame rate from 10ms to 40ms. Specifically, our implementation uses two convolutions with stride $(2, 2)$. To clarify the exact process, let's begin with the input of the subsampling layer — the LogMel Filterbank $\in \mathbb{R}^{B \times T_{10\text{ms}} \times 80}$, with $T_{10\text{ms}}$ being frames of 10ms

audio, stemming from the Short time Fourrier Transformation before, further explained in section 4.1. The general formulation for the effect of a convolution on the dimensionality of the input is the following[1]:

$$\begin{aligned} H_{\text{out}} &= \left\lfloor \frac{H_{\text{in}} + 2 \times \text{padding}[0] - \text{dilation}[0] \times (\text{kernel\_size}[0] - 1) - 1}{\text{stride}[0]} + 1 \right\rfloor, \\ W_{\text{out}} &= \left\lfloor \frac{W_{\text{in}} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) - 1}{\text{stride}[1]} + 1 \right\rfloor. \end{aligned} \tag{2.3}$$

Since we use neither dilation nor padding, this reduces to:

$$\begin{aligned} H_{\text{out}} &= \left\lfloor \frac{H_{\text{in}} - \text{kernel\_size}[0] - 1 - 1}{\text{stride}[0]} + 1 \right\rfloor, \\ W_{\text{out}} &= \left\lfloor \frac{W_{\text{in}} + \text{kernel\_size}[1] - 1 - 1}{\text{stride}[1]} + 1 \right\rfloor. \end{aligned} \tag{2.4}$$

And finally, with a stride of $(2, 2)$ and a kernel size of $(3, 3)$ as we use in our embedding layer, we end up with:

$$\begin{aligned} H_{\text{out}} &= \left\lfloor \frac{H_{\text{in}} - 3}{2} + 1 \right\rfloor, \\ W_{\text{out}} &= \left\lfloor \frac{W_{\text{in}} - 3}{2} + 1 \right\rfloor. \end{aligned} \tag{2.5}$$

Applied twice, we end up with a new temporal dimensionality:

$$T_{40\text{ms}} := \left\lfloor \frac{\lfloor \frac{T_{10\text{ms}} - 3}{2} + 1 \rfloor - 3}{2} + 1 \right\rfloor$$

Although we end up with this rather uninformative equation, importantly the distance between two frames ends up being 40ms because of the stride length. The minor details are due to the convolution at the beginning and end of the dimensionality. For the frequency dimension we end up with:

$$T_{\text{conv}} := \left\lfloor \frac{\lfloor \frac{80 - 3}{2} + 1 \rfloor - 3}{2} + 1 \right\rfloor = 19$$

Because the convolution create 256-dimensional vectors we end up with vectors of dimension $x \in \mathbb{R}^{B \times T_{40\text{ms}} \times 256 \times 19}$, which are then fed into a linear layer that transforms the vectors into vectors of shape $\hat{x} \in \mathbb{R}^{B \times T_{40\text{ms}} \times 256}$. To these, finally, the positional encoding (see next section) is applied and afterwards, they are fed through the Convolution Blocks.

Later, we will use this property of $T_{40\text{ms}}$ covering 40ms of audio to calculate the timing predictions with our cross-attention based algorithm.

[1] `https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html`

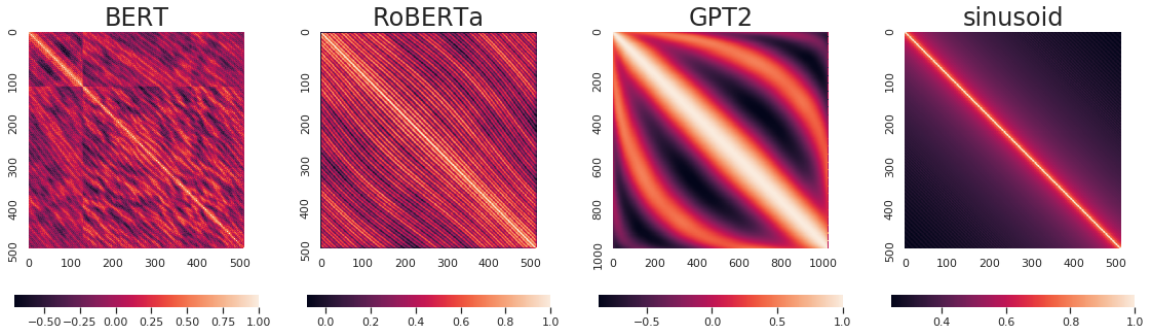


Figure 2.4.: Similarity of different positions, x- and y-axis, of different positional encodings [41].

2.3. Positional Encodings

In the attention mechanism, a weighted sum of context is computed. However, without the inclusion of positional encodings, there is no inherent information about the order of elements within the context. Various positional encoding strategies have been proposed, which can generally be categorized into learned and fixed encodings. Learned encodings, as the name suggests, are optimized during training and are therefore can also be referred to as learned positional embeddings.

In contrast, fixed positional encodings are statically applied to the input sequence. A simple approach, such as representing positions with natural numbers $n \in \mathbb{N}$, has notable limitations. These encodings increase in magnitude with growing sequence lengths, which can negatively affect computation, particularly because the attention mechanism relies on the dot product to calculate similarity. Moreover, this approach lacks an efficient mechanism for computing relative distances between positions.

The authors of [39] introduced sinusoidal positional encodings, which address these issues effectively.

Absolute positional encoding The most common absolute positional encoding is the sinusoidal positional encoding defined by the authors of [39] as:

$$PE_{(\text{pos, 2i})} = \sin\left(\frac{\text{pos}}{10000^{\frac{2i}{d_{\text{model}}}}}\right)$$

$$PE_{(\text{pos, 2i+1})} = \cos\left(\frac{\text{pos}}{10000^{\frac{2i}{d_{\text{model}}}}}\right)$$

Here, `pos` is the position, i.e., a natural number $\text{pos} \in \mathbb{N}$, `i` is the dimension `i` of the sinusoidal positional encoding vector $\text{PE} \in \mathbb{R}^{d_{\text{model}}}$, with d_{model} being the feature-dimension of the model. This dimensionality is purposefully chosen, as the positional encoding `PE` is added to the feature vector.

Even though this type of positional encoding encodes the absolute position, the authors chose it, because they believed the relative position can be easily calculated from it[39]. In fact, it can be shown, that for any position PE_{pos} a transformation to a fixed offset k, $PE_{\text{pos}+k}$ can be represented by a linear transformation [39, 17].

The sinusoidal positional encoding can be seen in the most right plot in Figure 2.4. In the same figure, learned positional embeddings from models `GPT2`, `RoBERTa` and `BERT` can also be seen. Most importantly, they also show a diagonal structure, meaning positions closer to each other have a higher similarity.

Relative positional encoding Although the authors of the sinusoidal positional encoding in [39] assume the possibility of extrapolation of these absolute positional encodings, it has been shown, that in practice this extrapolation does not work and the quality degrades quickly if the sequence length exceeds what the model has seen during training [29]. Furthermore, authors have shown that models using absolute positional encodings have different representations for identical sentences with shifted starting positions, which in turn affected task performance [33].

The framework ESPnet uses the relative positional encoding introduced in [6] by default, but also offers the absolute positional encodings introduced earlier. [6] use the same sinusoidal calculation as in the sinusoidal absolute positional encoding, however, instead of plugging in absolute positions, they use the definition for relative positions.

Specifically, they directly modify the calculation of the Attention:

$$A_{i,j}^{\text{abs}} = \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{E}_{x_i}}_{\text{(a)}} + \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{U}_j}_{\text{(b)}} + \underbrace{\mathbf{U}_i^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{E}_{x_i}}_{\text{(c)}} + \underbrace{\mathbf{U}_i^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{U}_j}_{\text{(d)}} .$$

where $\mathbf{E}_{x_i}$ is the word embedding of position `i` and $\mathbf{U}_j$ is the absolute positional embedding of position `j`. By using relative positional encodings, they replace $\mathbf{U}_i$ with learnable vectors $u^\top$ and $v^\top$ and end up with the following equation:

$$A_{i,j}^{\text{rel}} = \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_{k,E} \mathbf{E}_{x_i}}_{\text{(a)}} + \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_{k,R} \mathbf{R}_{i-j}}_{\text{(b)}} + \underbrace{\mathbf{u}^\top \mathbf{W}_{k,E} \mathbf{E}_x}_{\text{(c)}} + \underbrace{\mathbf{v}^\top \mathbf{W}_{k,R} \mathbf{R}_{i-j}}_{\text{(d)}} .$$

2.4. Encoder-decoder models and cross-attention

In our experiments, we use the speech recognition framework ESPnet[42]. Specifically, we chose an encoder-decoder architecture with a Conformer as encoder and a Transformer as decoder with a hybrid CTC-Attention loss. This choice allows us to leverage the inherent cross-attention in encoder-decoder models for EOU prediction.

2.4.1. Encoder-decoder Architecture

In ASR, the input is sampled audio data, and the output is the text transcript. In contrast to the sequence labeling task in Machine Learning, where there is a 1-to-1 relation between input and output — i.e., 1 output per input — ASR has to deal with differences in input and output length. However, in contrast to Machine Translation (MT) the input and output share the same order, whilst in MT different languages can lead to different sentence structures.

The encoder-decoder architecture shown in Figure 4.1 has an M-to-N relation, i.e., is able to handle any relation. This is achieved, by using a start token `<BOS>` and a stop token

<EOS> to begin and stop decoding dynamically. This way, the decoder can decide how many tokens to decode and emit the <EOS> token, whenever the decoding is complete.

The encoder encodes the information of the input and stores an intermediary hidden representation that the decoder can access during the decoding of the output. Different methods exist for the decoder to access the intermediary encoder information. However, the most common approach nowadays is cross-attention, which we use for our EOU prediction algorithm in our experiments.

Cross-attention is another use case of the attention mechanism. Contrary to the self-attention employed in the Transformer layers, for cross-attention the keys K and values V are computed from the encodings, i.e., hidden representations of the encoder that the decoder layer attends to. This is usually the last layer of each encoder step.

2.4.2. CTC monotonicity and Hybrid CTC-Attention Models

An alternative architecture frequently used in ASR systems is based on Connectionist Temporal Classification (CTC). CTC operates on the encoder and establishes a 1-to-1 relation between input and output frames. Despite this, CTC allows outputs of arbitrary lengths by introducing an empty token, ϵ, and collapsing consecutive identical tokens into a single token. For example, the output sequence "Caaaaarr" would be compressed into "Car". While CTC training, inference, and optimization involve several intricacies, which are not relevant for this thesis and thus omitted, its fundamental properties make it particularly advantageous for certain tasks.

The hybrid CTC-attention architecture employed in the ESPnet framework [43] combines the strengths of both CTC and attention-based approaches. As illustrated in Figure 2.5, the hybrid architecture applies the CTC loss on the encoder and the attention loss on the decoder. During decoding, it evaluates a combination of CTC and attention scores, weighted by a parameter α:

$$\log p^{\text{hyb}}(y_n \mid y_{1:n-1}, \mathbf{h}_{1:T'}) = \alpha \log p^{\text{ctc}}(y_n \mid y_{1:n-1}, \mathbf{h}_{1:T'}) + (1-\alpha) \log p^{\text{att}}(y_n \mid y_{1:n-1}, \mathbf{h}_{1:T'}).$$

This hybrid approach allows the decoder to benefit from the monotonic constraints of CTC while leveraging the flexibility of attention mechanisms.

One notable property of CTC training is its promotion of monotonic alignments [43], which can serve as a reference for other objectives, such as cross-attention alignment [16]. This monotonicity is highly relevant to our work, as it provides the motivation for the cross-attention decoding algorithm. Specifically, we hypothesize the monotonicity induced by CTC training on the encoder representations to guide the cross-attention mechanism to attend in a similarly monotonic manner. Our EOU prediction algorithm is based on the assumption, that this monotonicity extends to additional empty frames (which still posses positional encoding), which we add to the encoder.

In our experiments, the monotonicity induced by hybrid CTC training appeared sufficient for guiding the cross-attention mechanism. However, exploring additional methods to enhance this monotonicity continuation could be an interesting direction for future work. While we attempted various approaches, as detailed in section 5.3, these efforts did not yield successful results.

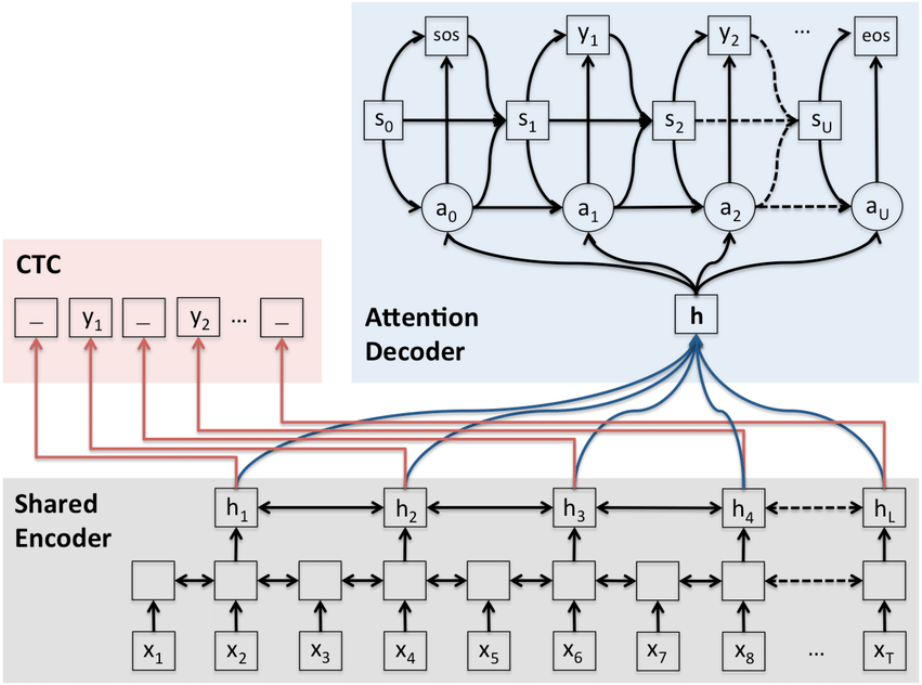


Figure 2.5.: Hybrid CTC-Attention architecture: The CTC loss is applied directly to the encoder, while the attention loss is propagated backward from the decoder.

2.5. Dialogue Systems

Dialogue systems are interactive systems that a (human) user can interact with. Ideally, the systems provide a natural, coherent and context-aware dialogue. Dialogue systems are usually made up of several modules, e.g., ASR module and Response Generation module, but could also include modules like Dialogue Management or Language Understanding.

Because humans are used to immediate responses dialogue systems are usually built in a streaming manner, i.e., processing inputs continuously to reduce latency, making the interaction with he Dialogue System more seamless. The intricacies of such streaming architectures are further described insubsection 2.5.1.

One crucial part of a dialogue system is to determine when the user has finished an utterance, after which the ASR transcribed input can be forwarded to the next module. This is usually referred to as EOU detection or endpointing, further elaborated in subsection 2.5.2.

In this work, we go one step further by predicting the last part of the spoken utterance, i.e., predicting the EOU and the final words leading to the EOU. This concept and the corresponding evaluation metrics are discussed in subsection 2.5.3.

2.5.1. Streaming Architectures

In offline ASR, the counterpart to streaming ASR, the whole audio is consumed to generate a transcript. However, in real-time systems, the streaming property is desired, i.e., continuously consuming input audio and generating the transcript step by step. Using

this approach, intermediary outputs can be generated, enabling more interactive communication with the dialogue system and reducing latency.

You are absolutely correct! Merely masking the attention matrix is not sufficient for a fully streaming architecture in a Conformer or Transformer-based encoder. To ensure that the model adheres to causal constraints, you also need to use **causal convolutions** in the convolutional modules of the Conformer encoder. This ensures that the convolutional layers only have access to past context, maintaining the streaming nature of the architecture.

The classical Transformer and Conformer encoders are not suited for a streaming architecture because, with standard training, the Attention mechanism has access to all future context. However, in a streaming dialogue system, the system only has access to previously uttered context. Thus, in our work, we modify the Conformer encoder to only have access to previous context, which is achieved through causal Attention and causal convolutions. Causal Attention is implemented by masking the Attention matrix, while causal convolutions ensure that the convolutional modules respect the same constraint.

A commonly used architecture for streaming ASR systems are Transducer based models [12][48]. However, these architectures do not provide a trivial way of doing EOU prediction. Thus, our architecture of choice is the causal Conformer encoder Transformer decoder architecture.

2.5.2. End-Of-Utterance Detection and Estimation

The authors of [9] define EOU as the point in time when the waveform collapses. Figure 2.6 shows a schematic drawing of such a waveform and the EOU indicated as t_{EOU}.

Having an accurate and correct EOU Detection/Prediction is crucial for a Dialogue System[28]. If the EOU prediction is too early, the user might get cut off and the subsequent system might not get sufficient information to process an optimal response. On the other hand, if the prediction is too late the latency will reduce user experience.

Another general latency reduction approach, is to lower computational latency, i.e., lower inference time. This can be achieved, by reducing the model size, which usually leads to a reduction in accuracy. Thus inference latency ends in a trade-off between accuracy and latency.

In this work, we intend to reduce latency in neither of the two ways described above. Instead, we want to do so, by predicting the EOU instead of detecting the EOU.

EOU Detection In the task of EOU Detection, given an acoustic input O, the goal is to determine whether or not the given audio contains the EOU and, if so, identifying the exact point of time. Importantly, O does not need to be a complete utterance, as most EOU Detection modules are used as part of a streaming dialogue system.

EOU Prediction In contrast to EOU Detection, there has been limited research conducted in this field so far, resulting in few formal definitions of the task. We define EOU Prediction as follows: Given an incomplete audio input $\hat{O}$, the objective is to estimate how far into the future, beyond the end of $\hat{O}$, the actual EOU occurs.

Common metrics for EOU Detection are:

- **EP cutoff**: How often the user is cut off.

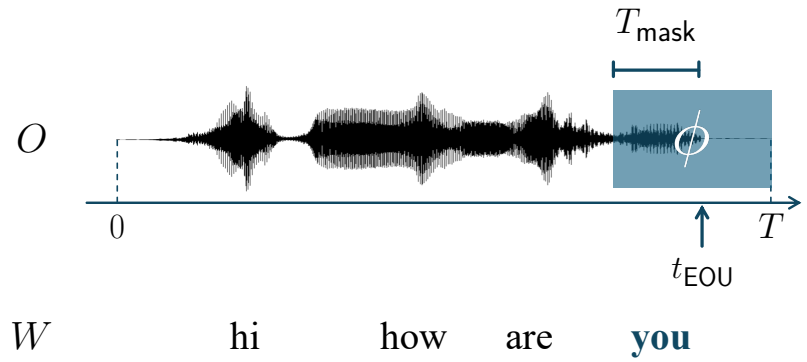


Figure 2.6.: Schematic drawing of predictive tasks of interest. Given an utterance of which we mask T_{mask} milliseconds ahead of EOU and the trailing silence shown as ϕ, predictive EOU detection tries to predict t_{EOU} based on the available audio information O. The goal of predictive ASR is to generate the masked words of the transcript W that correspond to the masked input (i.e., "you") based on the visible audio information and preceding tokens.

- **EP50 & EP90**: 50- and 90 percentile latency.
- **WER**: Word Error Rate influence by the EOU system.

In this work we will use boxplots to illustrate the EOU prediction performance. Specifically, the absolute difference between predicted end-of-utterance and ground-truth end-of-utterance is calculated as latency: $|\hat{t_{\text{EOU}}} - t_{\text{EOU}}|$, as it is the most straightforward representation of performance.

2.5.3. Future Word Prediction

Just like with EOU prediction, with Future Word prediction the goal is to reduce latency, i.e., give the subsequent module more time. Given an incomplete audio acoustic input O, the model's goal is to predict the complete transcript, including the transcript of the missing part of the audio. In the schematic example in Figure 2.6 this would include the masked word "you".

For evaluation of this predictive capability we introduce Future Word Error Rate (FWER). FWER is computed by using ground-truth tokens for the spoken content and calculating WER for the tokens predicted into the future. This allows for the exclusive measurement of future predictions, isolating them from the cumulative errors associated with the previously predicted tokens.

3. Related Work

Different research use different definitions for latency [21, 34]. For example, [34] consider themselves with **Average Token Emission Delay (AvgTD)**, the average delay between all emitted tokens and their respective timestamps from alignments. Others like [21] consider themselves with **prefetch latency**, defined as the time between the first correct prefetch (using a partial hypothesis to send an early request) and the end of the utterance.

In our work, when we refer to latency, we mean the latency between the actual end-of-utterance t_{EOU} and the point in time that the subsequent dialogue-system component is given to ASR input.

There are many ways, to achieve such general latency reduction. One way to achieve such latency reduction is AvgTD defined above. However, also trivial solutions, such as quantization for computional efficiency will lead to faster inference times and because of that to less latency. Even reducing the latency of the endpointing system, that determines the end of the utterance, will result in a reduction in overall latency.

Finally, with future word generation, the theoretical latency can be further decreased even to a negative latency, by predicting the words to be uttered.

Minimizing these latencies is desired for streaming ASR systems, as they impact user experience. Efforts to reduce these latencies can be grouped into two broad approaches: optimizing the latency of individual subsystems such as ASR and EOU detection (section 3.1, section 3.2), and leveraging future word generation to enhance prefetching and reducing the theoretical latency (section 3.3).

3.1. ASR latency reduction

Reducing ASR latency is the most trivial way to minimizing overall system latency. Since streaming architectures operate in real-time, much work has focused on limiting the future context required for decoding [25, 21, 24, 49, 44, 36]. This reduction, often referred to as limiting "look-ahead frames," improves latency but degrades ASR accuracy.

To address this trade-off, dual-encoder systems combining causal and non-causal encoders have been widely explored [49, 25, 44, 23]. Causal encoders operate with minimal latency, while non-causal encoders use more context to improve accuracy. These encoders can be arranged in cascaded [25], parallel [24], or knowledge-distilled configurations [44, 24].

Another approach is chunking, where the audio stream is divided into fixed-size segments for processing [7, 32, 38, 4]. Chunking balances latency and accuracy by limiting the effective receptive field of the model while maintaining manageable computational complexity.

Additionally, [36] introduce schedulers to dynamically regulate look-ahead context in transducer models, by introducing a regularization loss. Using this method, they achieve comparable WER to full-context models with almost streaming level latency.

Emission Time Reducing the emission time of the decoder, i.e., the previously mentioned AvgTD, is another strategy for latency reduction in the context of a streaming ASR system. FastEmit [45] biases RNN-T models to emit tokens more quickly by penalizing blank predictions. Similarly, TrimTail [34] introduces a training strategy in which parts of the input spectrogram are randomly trimmed, encouraging earlier token alignment.

[24] demonstrate that conformer-based encoders suffer from longer emission latency compared to traditional transformers, while also noting the benefits of their dual causal/non-causal architecture for emission timing.

3.2. EOU Detection

EOU detection is critical to minimize endpointing latency, which directly impacts response timing. Traditional approaches often rely on Voice Activity Detectors (VADs), which are limited by their lack of semantic understanding. To address this, [31] proposed LSTM-based models to incorporate linguistic context.

Recent work has focused on architectural optimizations for real-time performance. For instance, [9] introduce a binarized state sequence for use with force alignment algorithms. By simplifying the sequence representation, they achieve faster inference times while maintaining accuracy.

Research has also been conducted to automatically tune the "aggressiveness" of the endpointing system. A more aggressive system will predict a EOU earlier but also cut off the user more often. A less aggressive system will predict EOU less often and lead to higher latency. [35] use a reinforcement learning approach to choose an optimal configuration.

In addition, multimodal systems have gained attention. For example, [30] integrate semantic information from an external language model to improve EOU detection with added semantic completeness information. However, reliance on external LMs can add computational complexity.

Our approach differs from these methods. We use a cross-attention based algorithm in a single model. This saves additional computational latency while also working with EOU Prediction instead of EOU Detection to further decrease theoretical latency.

3.3. Future word generation

Predicting future tokens can be utilized both for EOU prediction and ASR. [30] use an external LM to generate probabilities of an utterance ending in the next k tokens. They use this semantic completeness information for EOU detection. Another approach is to introduce a turn taking token into a LLM [8], which can then be used to predict EOUs.

For ASR, [46] use an external LLM and prompt it with the incomplete audio context to generate the rest of the sentence.

While effective, these methods (except for [46]) depend heavily on external language models, which do not incorporate acoustic information. In contrast, our method avoids external LMs and thus is able to include acoustic information such as prosody and information on partially spoken words. We do so, by training our encoder-decoder model to predict future tokens directly using a masked training approach.

4. Methodology and Setup

This chapter outlines the methodology and experimental setup used to develop an ASR system with predictive capabilities, avoiding reliance on an external language models. Our goal is to keep the architecture self-contained by employing a single model, by just introducing modifications to the training process, enhancing the system's intrinsic language modeling capabilities.

To achieve this, we use a standard encoder-decoder model and incorporate a masked training strategy. This approach lets the model predict tokens directly. We believe, this strategy enhances the models predictive functionality while retaining acoustic awareness. Additionally, we leverage the cross-attention mechanism within the encoder-decoder architecture to facilitate End-of-Utterance (EOU) prediction, a critical feature for dialogue systems.

The masked training strategy involves masking the end of the audio input, forcing the decoder to rely on the acoustic information encoded in the remaining input to predict future tokens. This enhances the model's ability to make predictions and integrates predictive functionality into a conventional ASR framework.

This chapter is structured as follows: section 4.1 section 4.2 details the training and inference methodologies, section 4.3 discusses the evaluation methods, section 4.4 describes the datasets and preprocessing steps, and section 4.5 presents the experimental setup, including hardware, software, and model configurations.

4.1. Task formulation

In order to define our desired tasks, let us begin by defining the given raw audio input O as a sequence of T_s samples:

$$O = [o_1, o_2, \ldots, o_{T_s}],$$

where:

- o_i is the amplitude of the audio signal at the i-th time step.
- T_s is the total number of samples in the audio signal, given by:

$$T_s = \text{Sampling Rate} \times \text{Duration}.$$

In our case the audio is either sampled or up-sampled to 16kHz. Through a series of preprocessing — including Short time Fourrier Transformation (STFT) — O gets processed to a Log-Mel-Fbank, i.e, a logarithmic Mel Filterbank $O \in \mathbb{R}^{B \times T_{10\text{ms}} \times 80}$, with B being the batch-size and $T_{10\text{ms}}$ being the time in 10ms steps, i.e., $T/10$ with T being the total time

in ms. Because we choose `win_length = 400` and `hop_length = 160` as parameters for the STFT, we end up with each each element $o \in O$ covering 10ms of audio input with 80 features each.

The corresponding transcript $\mathcal{W}$ is a sequence of subword units:

$$\mathcal{W} = [w_1, w_2, \ldots, w_N],$$

where:

- $w_i \in \mathcal{V}$ is the i-th subword unit in the transcript, and $\mathcal{V}$ is the vocabulary of subword units.
- $|\mathcal{V}|$ denotes the size of the subword vocabulary.
- N is the total number of subword tokens in the transcript.

Notably, for `SWBD` $|\mathcal{V}| = 2000$ and for `LS100` $|\mathcal{V}| = 5000$.

For simplicity, we assume the log-Mel features O, instead of the raw audio $\mathcal{O}$ is the input of the model and see the pre-processing as given instead of as part of the model.

We want our model to be able to predict and transcribe the transcript $\mathcal{W}$ using partial information of the log-Mel features $\hat{O}$ of the full log-Mel features O, i.e.:

$$\hat{O} := \Big[o_1, \ldots, o_{t_{\text{EOU}}-k-1}, \underbrace{\begin{pmatrix}0\\\vdots\\0\end{pmatrix}_{t_{\text{EOU}}-k}, \ldots, \begin{pmatrix}0\\\vdots\\0\end{pmatrix}_{t_{\text{EOU}}}}_{k \text{ zero-vectors}}, \underbrace{\begin{pmatrix}0\\\vdots\\0\end{pmatrix}_{t_{\text{EOU}}+1}, \ldots, \begin{pmatrix}0\\\vdots\\0\end{pmatrix}_{T_{10\text{ms}}}}_{\text{remaining vectors are zero-vectors}}\Big]$$

We chose this formulation of $\hat{O}$ that has the remaining vectors set to zero-vectors as well, instead of setting the last k frames of the audio directly to zero-vectors, because it allows for a better statement on how much masked **speech** our model can speculatively generate. For both training and evaluation, $\hat{O}$ is derived from O by setting the last k log-Mel feature vectors preceding the EOU (t_{EOU}) and all subsequent vectors until the end of the audio to zero. Specifically, this means that we set the zero-vectors after the pre-processing to log-Mel features and before the embedding layer `embed` of the conformer encoder. The embedding layer contains the following neural network layers:

```
(embed): Conv2dSubsampling(
    (conv): Sequential(
        (0): Conv2d(1, 256, kernel_size=(3, 3), stride=(2, 2))
        (1): ReLU()
        (2): Conv2d(256, 256, kernel_size=(3, 3), stride=(2, 2))
        (3): ReLU()
    )
    (out): Sequential(
```

```
        (0): Linear(in_features=4864, out_features=256, bias=True)
        (1): PositionalEncoding(
            (dropout): Dropout(p=0.1, inplace=False)
        )
    )
)
```

We want to highlight, that the absolute positional encoding is applied during this encoder embedding after the convolutional subsampling, which is explained in detail in section 2.2.

In practical system usage during inference, zero-vectors would need to be appended to the incomplete audio O to construct $\hat{O}$. This requirement also assumes that a prior decision must be made, determining that the EOU is near.

Predictive ASR For predictive ASR, we want the model to be able to use incomplete audio $\hat{O}$, in order to predict the corresponding transcript $\mathcal{W}$. To do this, we use the standard decoding algorithms typical of encoder-decoder-based ASR models. Specifically, we want to minimize the FWER metric.

EOU Prediction For EOU prediction, the model also gets $\hat{O}$ as input. The goal is to minimize the difference between the predicted EOU $t_{\hat{EOU}}$ and the actual EOU t_{EOU}. For the prediction of $t_{\hat{EOU}}$, we introduce a cross-attention based algorithm introduced in the following section.

4.2. Training and Inference

4.2.1. Inference

EOU prediction For EOU prediction we use our proposed cross-attention mechanism, depicted in Figure 4.1. Given an incomplete audio input $\hat{O}$, the encoder generates audio representations $H \in \mathbb{R}^{T_{40\text{ms}} \times D_{\text{model}}}$, and, subsequently, the decoder produces token representations $Q \in \mathbb{R}^{(N+1) \times D_{\text{model}}}$. Here, D_{model} denotes the dimensionality of the hidden layers within each network. Notably, the $(N+1)$-th output from the decoder is specifically dedicated to predicting the end-of-sentence token — `<EOS>`. During the computation of Q, the model computes a cross-attention score matrix $\mathbf{A} \in [0,1]^{(N+1) \times T_{40\text{ms}}}$ by applying the scaled dot-product attention against H. Given the attention scores $\mathbf{a}_{N+1} \in \mathbf{A}$ for the `<EOS>` token, the maximum score in $\mathbf{a}_{N+1}$ is defined as $a_{\max} = \max(\mathbf{a}_{N+1})$. Finally, the EOU time is estimated based on $a_{\max}$ as

$$\hat{t}_{\text{EOU}} = \tau \cdot \max\{t \mid a_t \geq \Psi \cdot a_{\max}\}, \tag{4.1}$$

where τ represents the duration of each encoder frame (i.e., $\tau = 40$ ms, see Equation 2.2), and the hyperparameter Ψ is introduced to threshold the upper limit of the frames that receive attention. Ψ is fine-tuned on the development sets of *LS*100 and *SWBD* respectively.

For example in Figure 4.1 with $T_{40\text{ms}} = 8$ and $N = 3$, the attention scores in $\mathbf{a}_4$ can extend across the encoder outputs. Since our goal is to identify the EOU, we seek the rightmost

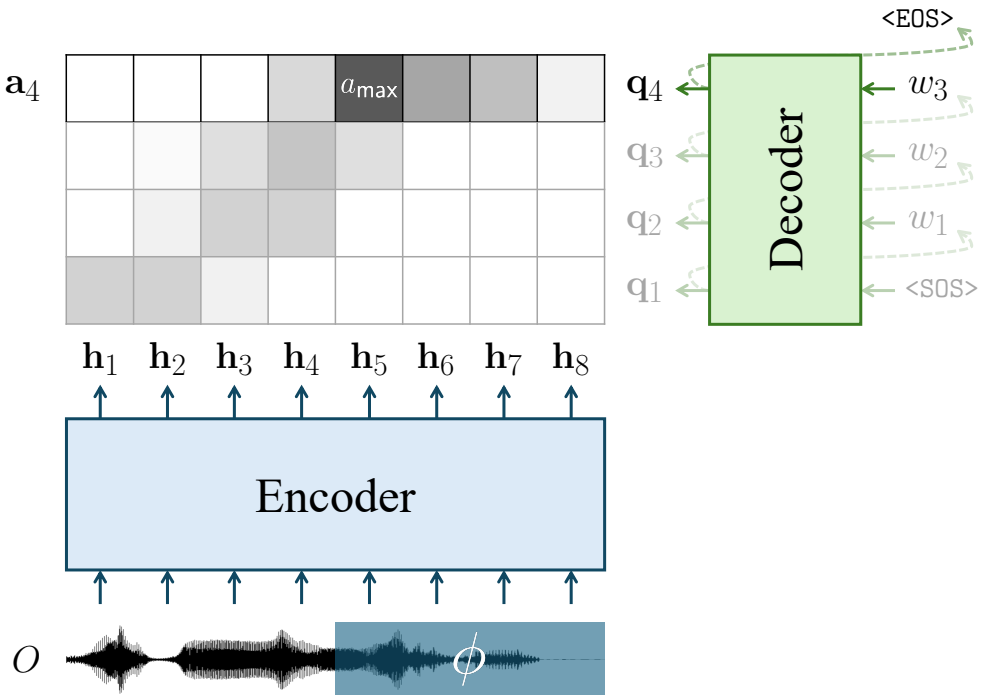


Figure 4.1.: Proposed approach for detecting EOU based on cross-attention mechanism. It computes attention scores used for generating the final output, the end-of-sentence token (<EOS>). To identify the EOU, the upper boundary of the frames related to this final output is determined by comparing scores $a_t \in \mathbf{a}_4$ to the maximum score a_{max}.

frame associated with $\mathbf{q}_4$ based on the maximum score a_{max}, where the estimated EOU $\hat{t}_{\text{EOU}}$ is likely to be at frame 6, 7, or 8, depending on the value of Ψ.

The above algorithm can work whether or not future content in the audio input is available. When the entire input is accessible, it performs straightforward EOU detection. On the other hand, in cases where future input is absent, it becomes the predictive EOU task, which is our primary focus.

Predictive ASR During predictive ASR, the model gets the incomplete audio $\hat{O}$ as input and standard ASR decoding is performed, i.e., the decoder is prompted with the <SOS> token to begin decoding and is decoding until the appearance of the <EOS> token.

For FWER results, we use the word-level alignments to determine, which words are partially or completely masked during inference. Afterwards, we prompt the decoder with all tokens that are neither partially nor completely masked. This is done, to avoid accumulated ASR errors influencing the prediction results.

For our hybrid CTC-attention architecture we use `ctc_weight = 0.3` with a beam-size of 1, i.e., greedy search, for our WER results. For the n-best FWER results, we use a beam-size of 20 with `ctc_weight = 0.0`. The latter is chosen, because otherwise we encountered repeats during decoding.

4.2.2. Training

To address the predictive tasks, we design a training that is explicitly gives the decoder an incentive to increase its acoustically conditional language model probabilities. Specifically, we use the log-Mel features of the partial audio $\hat{O}$ as input for the model, as illustrated in Figure 2.6. The masked duration $T_{\text{Mask}} = k \cdot 10$ is uniformly sampled, so that the model generalizes on different masking duration during training. To address variations in the duration of silence after t_{EOU}, as indicated in Figure 4.2, we also introduce variability in the audio input length. This is achieved by sampling a duration k_{add} from a uniform distribution between $-T_\Delta$ and T_Δ and accordingly adjusting the length of the masked input by adding or removing zero vectors. If a positive value k_{add} is sampled, then the result is the following:

$$\hat{O} := \Big[o_1, \ldots, o_{t_{\text{EOU}}-k-1}, \underbrace{\begin{pmatrix}0\\\vdots\\0\end{pmatrix}_{t_{\text{EOU}}-k}, \ldots, \begin{pmatrix}0\\\vdots\\0\end{pmatrix}_{t_{\text{EOU}}}}_{k \text{ zero-vectors}}, \underbrace{\begin{pmatrix}0\\\vdots\\0\end{pmatrix}_{t_{\text{EOU}}+1}, \ldots, \begin{pmatrix}0\\\vdots\\0\end{pmatrix}_{T_{\text{EOA}}}}_{\text{remaining vectors are zero-vectors}}, \underbrace{\begin{pmatrix}0\\\vdots\\0\end{pmatrix}, \ldots, \begin{pmatrix}0\\\vdots\\0\end{pmatrix}}_{k_{\text{add}} \text{ additional zero-vectors}}\Big]$$

with T_{EOA} being the time of the end of the audio before adding extra frames. If a negative value is sampled, then the result is the following:

$$\hat{O} := \Big[o_1, \ldots, o_{t_{\text{EOU}}-k-1}, \underbrace{\begin{pmatrix}0\\\vdots\\0\end{pmatrix}_{t_{\text{EOU}}-k}, \ldots, \begin{pmatrix}0\\\vdots\\0\end{pmatrix}, \ldots, \begin{pmatrix}0\\\vdots\\0\end{pmatrix}_{t_{\text{EOU}}}}_{k \text{ zero-vectors}}, \underbrace{\begin{pmatrix}0\\\vdots\\0\end{pmatrix}_{t_{\text{EOU}}+1}, \ldots, \begin{pmatrix}0\\\vdots\\0\end{pmatrix}_{T_{\text{EOA}}}}_{\text{remaining vectors are zero-vectors}}\Big]$$

$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad}_{k_{\text{add}} \text{truncated zero-vectors}}$$

This training is expected to force the decoder to predict future tokens based on incomplete or absent acoustic information.

Importantly, we maintain positional encoding on the masked segment to assist the decoder in estimating the placement of future tokens relative to the unmasked part.

4.3. Evaluation

For our evaluation we compare two different models on both datasets SWBD and LS100 on the metrics FWER and EP50, as introduced in subsection 2.5.2 and subsection 2.5.3. Both models essentially share the same architecture, as described in section 4.5. However, we train one model, which we refer to as “baseline” without the masked training, as discussed in subsection 4.2.2 and compare it to the proposed model trained with masked training.

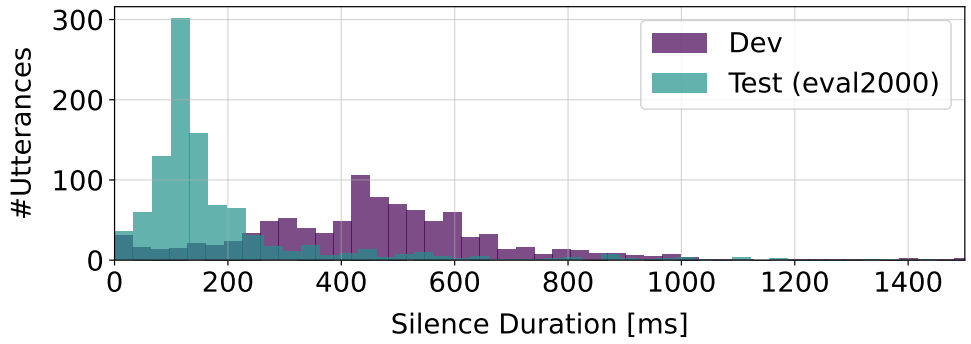


Figure 4.2.: Distribution of silence duration $T - t_{\text{EOU}}$ across development and test sets of Switchboard.

4.4. Datasets and Pre-processing

In this thesis, we use SWBD and LS100 as datasets for our experiments. Given the primary aim of this thesis—to contribute to dialogue systems development—we choose SWBD as dataset, as it contains naturally spoken dialogue over telephones. Specifically, participants are asked to talk about a specific topic in a 1-on-1 telephone conversation.

Additionally, we use the LS100 dataset, even though it consists of read book recordings, for it's ease of use and higher quality audio, for comparative analysis within our models.

SWBD The SWITCHBOARD (SWBD) [11] dataset transcripts contain noises markers, e.g., [noise], [vocalized-noise] or [laughter]. This can also result in utterances, solely containing noise markers. We remove these noises from the transcript and remove all utterances that only consisted of these. This results in a removal of 915 utterances out of 192645 utterances, amounting to only 0.5% of total utterances.

SWBD presents a more challenging scenario due to the dialogic nature, involving complex turn-taking between speakers with ambiguous endpoints. However, compared to librispeech, these utterances are almost always short enough, so no further splitting into shorter utterances is needed.

The following is a sample of the SWBD dataset. The full conversation is shown in section A.3:

```
B: this is archie
A: hi archie i'm sharon
B: have you seen dancing with wolf
A: yeah i've seen that that's uh that was a really good movie probably one of the
↪  best things about it was the scenery and uh
A: i thought the story was pretty good too i th- i think kevin costner did a
↪  really good job with it
B: have you ever lived in that part of the country
A: no i haven't
B: have you ever visited it
A: um i've visited the wyoming area i'm not sure exactly where dances with wolves
↪  was filmed
```

```
B: i think it was the black hills of south dakota
A: could be i- i haven't been to south dakota
A: ha- have you been up to that
B: well i lived in omaha for five winters and that rolling kind of uh
A: oh okay
A: terrain
```

LS100 The LibriSpeech [26] corpus contains 1000 hours of read English speech from audio-books as part of the LibriVox project. It contains two different development and test splits, "clean" and "other". "clean" is cleanly spoken speech, while "other" contains more challenging speech conditions. For our experiments, we restrict our usage to the "clean" test and development splits, and training is conducted solely on the 100-hour clean subset, "train-clean-100" (LS100).

The LibriSpeech dataset contains read passages, that are split into shorter segments. Longer segments are split into smaller ones, if a silence longer than 0.3sec exists [26]. This results in less natural end of sentences, compared to the natural turn taking manner of dialogues in SWBD. This can be seen as an approximation of the ground-truth of natural end-of-utterances.

The following is a snippet from a segment of the "dev-clean" dataset:

```
1272-128104-0000 MISTER QUILTER IS THE APOSTLE OF THE MIDDLE CLASSES AND WE ARE
↪  GLAD TO WELCOME HIS GOSPEL
1272-128104-0001 NOR IS MISTER QUILTER'S MANNER LESS INTERESTING THAN HIS MATTER
1272-128104-0002 HE TELLS US THAT AT THIS FESTIVE SEASON OF THE YEAR WITH
↪  CHRISTMAS AND ROAST BEEF LOOMING BEFORE US SIMILES DRAWN FROM EATING AND ITS
↪  RESULTS OCCUR MOST READILY TO THE MIND
1272-128104-0003 HE HAS GRAVE DOUBTS WHETHER SIR FREDERICK LEIGHTON'S WORK IS
↪  REALLY GREEK AFTER ALL AND CAN DISCOVER IN IT BUT LITTLE OF ROCKY ITHAC
...
```

Force alignment For both datasets the temporal alignments of the development set are used to optimize hyperparameters and the temporal alignments of the test set are used to evaluate the EOU predictions.

Since the LibriSpeech dataset does not provide native temporal alignments, we generated them for both the development and test sets using the Montreal Forced Aligner (MFA) [22]. The LS dataset features utterances with clear endpoints, making it well-suited for model evaluation under ideal conditions.

For the Switchboard (SWBD) dataset, although timestamp annotations are available for transcripts in the training set, the development set lacks such temporal information. To address this, we again utilized the MFA to produce temporal alignments for the SWBD development set.

We compared a CTC-based alignment[1] [18] with the Montreal Forced Aligner (MFA)[22] alignments. As seen in Figure 4.3, the MFA alignments align closer to the ground truth

[1] https://pytorch.org/audio/stable/tutorials/forced_alignment_tutorial.html

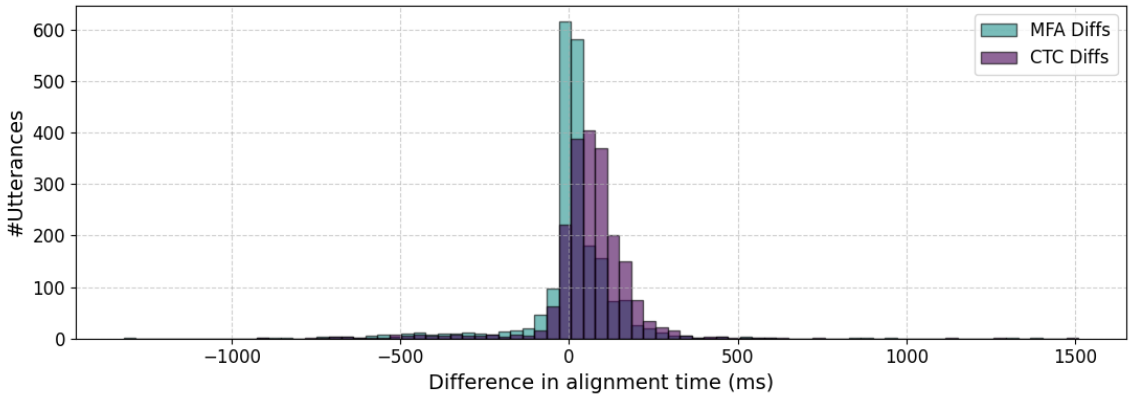


Figure 4.3.: Comparison of CTC and MFA alignments with the ground-truth alignments of the SWBD development dataset.

alignments of the SWBD development dataset. We thus chose to use the MFA for all alignments.

After the input is transformed into log-Mel Filterbanks, SpecAug[27] is used for data augmentation.

4.5. Experimental Setup

This section serves to describe experimental details, in order for the reader to be able to reproduce the experimental results. This covers, soft- and hardware, model architecture and hyper-parameters.

Architecture and Hyperparameters For all experiments, we use a slightly modified version of one of ESPnet's default recipes for the LS100 dataset. Specifically, this is an encoder-decoder architecture with a conformer, with 12 encoder layers, as encoder and a transformer, with 6 decoder layers, as decoder. The exact configuration can be found in section A.1 and a model summary of pytorch in section A.2. The following are the most essential aspects of the configuration:

The conformer is made to have causal attention, denoted by `is_causal: True`, by applying an attention mask. The model is a hybrid CTC-Attention model with the CTC weight set to 0.3, denoted by `ctc_weight: 0.3`.

In total the model consists of $33.44M$ trainable parameters. We use the Adam optimizer with a lr of $1.\overline{3} \cdot 10^{-7}$ and a warmup-lr of 0.002 with 15000 warmup steps for LS100 and 25000 for SWBD. We train for 70 epochs (LS100) and 100 epochs (SWBD) and average the 10 best checkpoints.

Soft- and Hardware For all experiments, we use the ESPnet[2] framework, which internally uses PyTorch[3] and Kaldi[4]. ESPnet is an open-source framework used for training speech processing neural networks.

[2]https://github.com/espnet/espnet
[3]https://github.com/pytorch/pytorch
[4]https://github.com/kaldi-asr/kaldi

We train our models split on 5 NVIDIA GeForce GTX 1080 GPUs on CUDA version 11.2 and PyTorch version 1.13.1. Each GPU has 11178MiB memory.

5. Experiments

In our primary experiments, we compare a baseline model, without masked training, vs. the proposed model with masked training.

In section 5.3, we also give a concise overview of some of our discontinued approaches.

5.1. EOU Prediction

LS100 Figure 5.1 reports box plots for LS-100 for both the baseline and the proposed model, showing the performance of our EOU detection using the cross-attention mechanism, which was measured by the absolute difference between the predicted and ground-truth EOU timing $|\hat{t}_{\text{EOU}} - t_{\text{EOU}}|$ (as detailed in subsection 2.5.2).

When $T_{\text{mask}} = 0$, indicating that the models had full access to the audio input, EOU was predicted reasonably well, with an average discrepancy of about 0 ms for the proposed model and about 20 ms for the baseline model. As the mask duration increases, the error for the baseline model increases notably, reaching an average difference of approximately 300 ms when $T_{\text{mask}} = 500$.

In contrast, the proposed model successfully mitigates this degradation especially for the higher mask durations, suppressing the average discrepancy to around 100 ms at $T_{\text{mask}} = 500$. This demonstrates the proposed training strategy, which involves masking future input, was effective in enabling the model to perform the predictive EOU detection task.

SWBD Figure 5.2 shows results on SWBD, following a similar trend as observed in LS-100 between the baseline and proposed models. However, the overall performance was inferior to LS-100, with greater variance in predictions. This indicates increased challenges in forecasting the EOU in conversational speech, where speaker terminations may be less distinct. Nonetheless, the proposed model consistently outperformed the baseline, particularly at the mask durations above 200 ms.

Analysis During analysis, we noticed that the baseline model generally decodes less tokens than the proposed model. We hypothesize, that the proposed model is able to decode the amount of tokens, that are masked, while the baseline model stops decoding directly, after reaching masked input. We first investigate the amount of masked words for our inference runs and compare to the amount of additionally decoded tokens of the proposed model.

Let a word alignment $w_{\mathcal{A}}$ be defined as a tuple $w_{\mathcal{A}} := (w_{start}, w_{stop})$. We define a word as **fully masked** if the masking threshold $\mathcal{M}_{\mathcal{T}} := t_{\text{EOU}} - T_{\text{mask}}$ is less than the start time of the word, i.e., $\mathcal{M}_{\mathcal{T}} < w_{start}$.

Similarly, a word is defined as **partially masked** if the masking threshold satisfies $w_{start} \leq \mathcal{M}_{\mathcal{T}} < w_{stop}$.

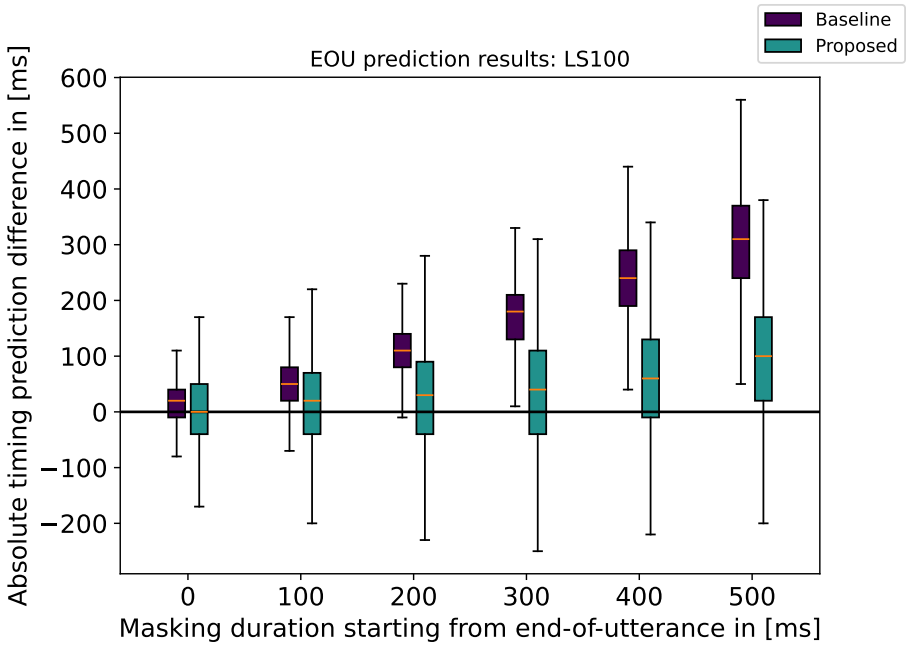


Figure 5.1.: The prediction results of the LS100 dataset.

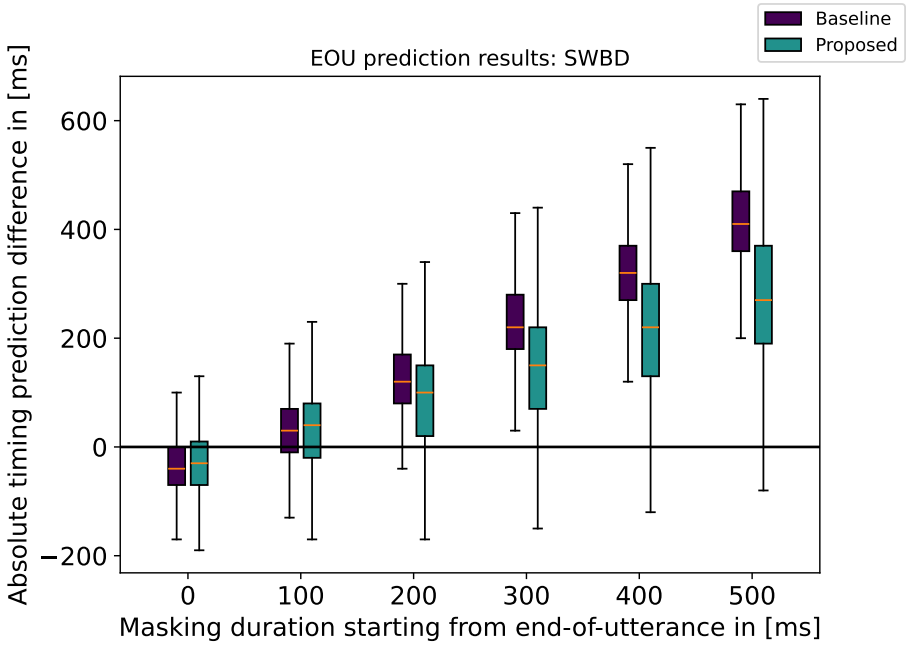


Figure 5.2.: The prediction results of the SWBD dataset.

Table 5.1.: Masked word statistic by mask duration [ms].

	Mask Duration T_{mask} [ms]					
Type	0	100	200	300	400	500
Fully masked	0	3	48	189	621	1476
Partially masked	0	2621	2657	2786	3180	3979

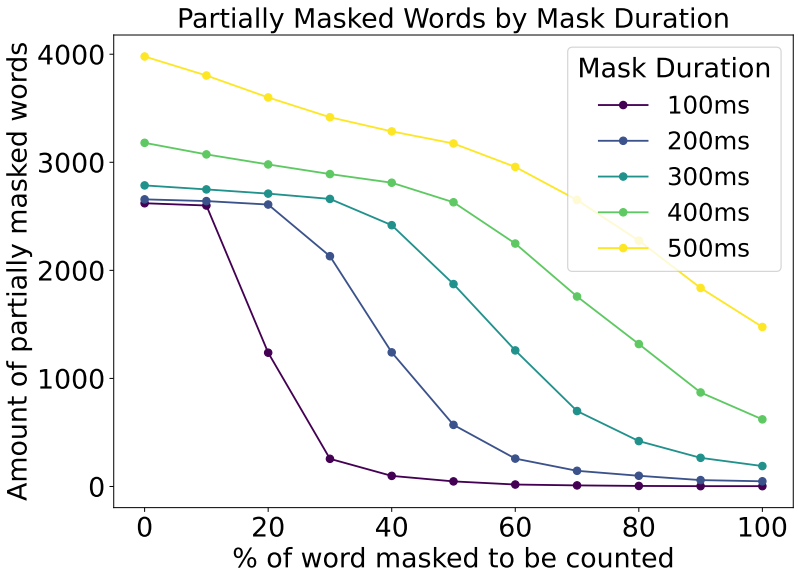


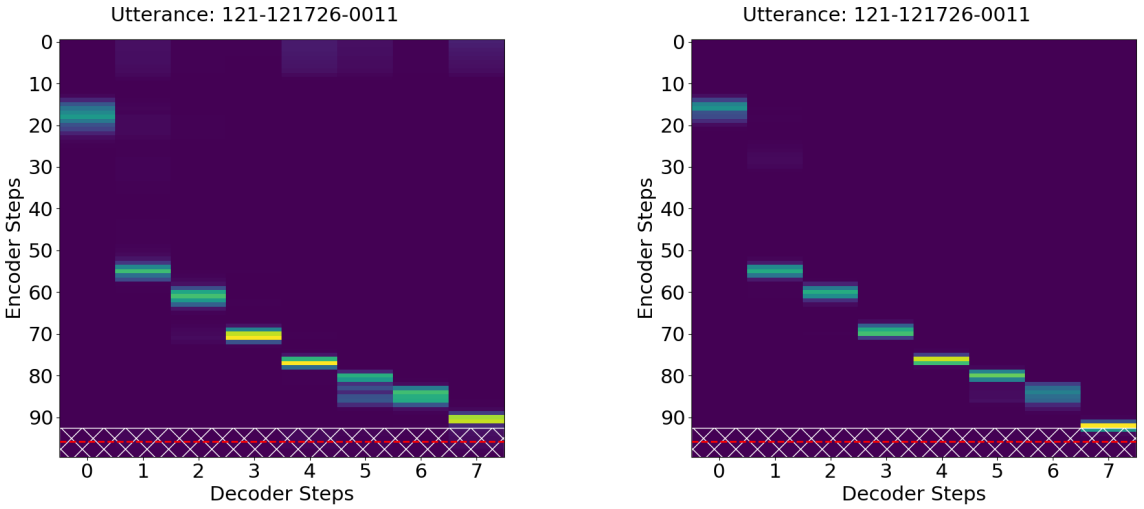
Figure 5.3.: Number of partially masked words (y-axis) based on the percentage of the word (x-axis) that must be masked to classify it as masked.

Table 5.1 shows how many words are partially and fully masked, when $\{0, 100, ..., 500\}$ms of audio before t_{EOU} are masked. We observe that up until 300 ms most words are only partially masked, with about 189 words of the 2620 utterances in the `test-clean` subset being fully masked.

To gain further insight into the distribution of partially masked words, in Figure 5.3, we can observe the number of partially masked words, based on how much of the word in [%] has to be masked in order for the word to be counted as partially masked. For simplicity, we will focus on the case $T_{\text{mask}} = 300$ for the remainder. We can see, that most words are between 30% and 70% masked and a total number of 2657 words being partially masked.

We find a similar number of words — 3569 — to be decoded less from the baseline model, compared to our proposed model. With the cross-attention monotonicity, as seen in Figure 5.4 in mind, this leads us to hypothesize, that a significant part of the better EOU prediction is based on more tokens being decoded and thus cross-attention being more attentive to encoder frames closer to t_{EOU}.

However, we also notice, that the cross-attention distribution differs, for the same amount of decoded tokens. In Figure 5.4, we see the cross-attention plots for the baseline and proposed model of the LS100 `test-clean` dataset. While we don't have a metric to



(a) Cross-attention plot of the baseline model.

(b) Cross-attention plot of the proposed model.

Figure 5.4.: The cross-attention of the baseline (left) and proposed model (right) of the same input utterance, 121-121726-0011 of the LS100 `test-clean` dataset. Even though both decoders decode the same amount of tokens, the cross-attention distribution differs. Furthermore, the baseline model is able to use acoustic information to correctly predict the last word `wife`, while the baseline model falsely predicts `what` as the last word. The ground truth text is: *HUSBAND THE NEXT THING TO A WIFE.*
The red horizontal line indicates the correct EOU timing t_{EOU}. The white dashed boxed indicates masked encoder input.
It seems like, the proposed model shows slightly more cross-attention attendance towards the masked area.

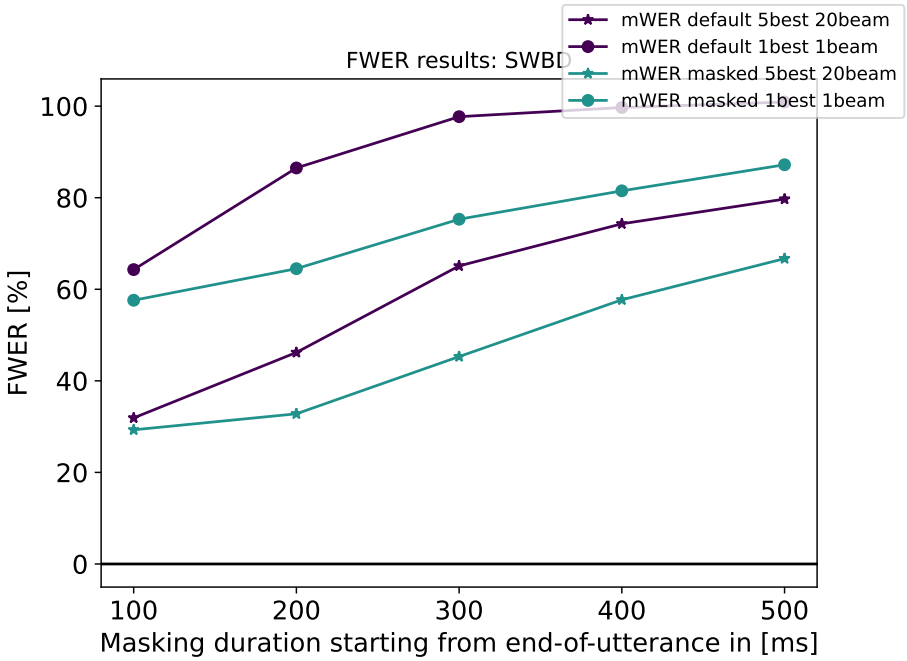


Figure 5.5.: The FWER predictive ASR results of the SWBD dataset.

proof it, we can only hypothesize based on looking at the attention-plots. We notice more attendance to encoder frames closer to t_{EOU} for the proposed model.

5.2. Predictive ASR

Figure 5.6 and Figure 5.5 plot the FWER results on LS-100 and SWBD, respectively, which assesses WER solely based on future predictions. We also present the FWER-at-5 (FWER@5) results, obtained by performing beam search decoding to generate the top-five hypotheses and reporting the lowest FWER observed among these.

Notice that the error rates are generally high, and this underscores the challenges of predicting upcoming tokens with various possible outcomes, consistent with the findings reported in [46].

Notably, the baseline models struggled with FWERs exceeding 80% for mask durations longer than 300 ms. In contrast, our model effectively reduces these errors thanks to the proposed training strategy, which we hypothesize, enhances the decoder's capability to act as a generative language model, even in the absence of audio input.

By evaluating FWER@5, our model greatly improved performance, suppressing errors to below 70%; however, we note that this comes at the cost of requiring the NLP modules to handle responses for five potential ASR hypotheses.

Table 5.2 reports the WER for LS-100 and SWBD, computed for all words (not limited to future words) predicted by the models. With the training strategy based on masking future input, the proposed model consistently outperformed the baseline across various mask durations. As we used the same recipe as ESPnet for LS100, we achieved similar

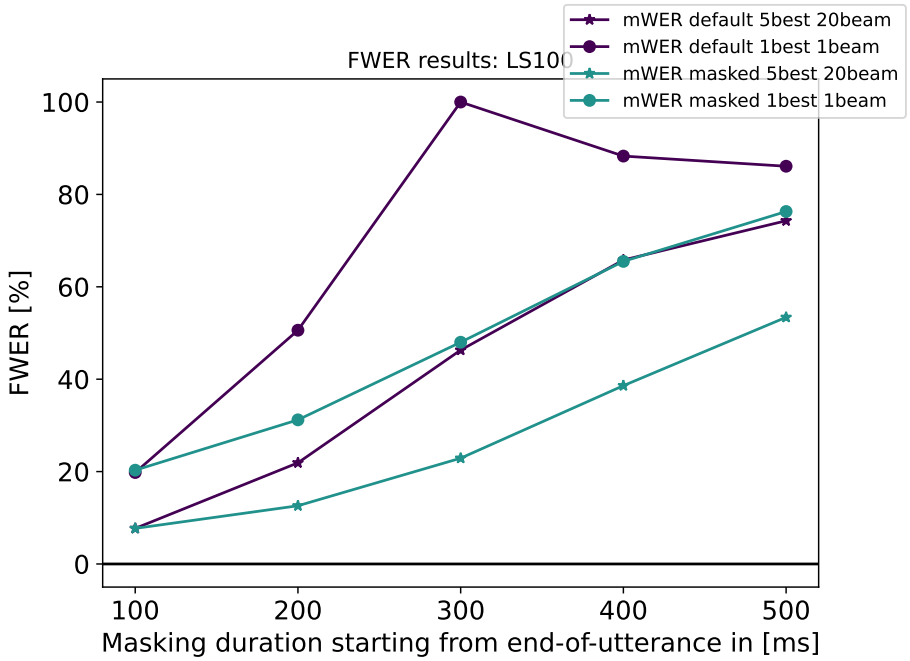


Figure 5.6.: The FWER predictive ASR results of the LS100 dataset.

Table 5.2.: WER [%] results on test sets for LS-100 and SWBD. WER results contain all decoded tokens and is not limited to future words like FWER.

		Mask Duration T_{mask} [ms]					
Dataset	**Model**	0	100	200	300	400	500
LS-100	Baseline	8.5	8.8	10.5	12.3	13.4	14.3
	Proposal	**8.4**	**8.6**	**9.2**	**10.3**	**11.7**	**13.2**
SWBD	Baseline	30.9	31.8	35.3	38.8	41.3	43.6
	Proposal	**30.5**	**30.1**	**32.2**	**34.9**	**37.8**	**40.2**

results, i.e., they achieve 8.4 WER[1] and our baseline model achieves 8.5 WER for $T_{\text{mask}} = 0$ The same comparison can not be made for SWBD, since we use the same architecture, that we used for LS100 for SWBD too, but ESPnet does not provide results for that recipe.

Interestingly, the proposed model decreased WER even for $T_{\text{mask}} = 0$, i.e., standard ASR, for both SWBD and LS100. We report a decrease of 0.1 WER for LS100 and a decrease of 0.4 WER for SWBD. We disregard 0.1 WER of LS100 as non-significant, however the additional minor improvements on SWBD let us believe, that this improvement could be attributed to enhancements in the decoder's ability to act as a language model, which could have aided in learning dependencies among output tokens.

Analysis In order to better understand the effects of the proposed training method on the predictive capabilities of the model, we evaluate the prediction accuracy of the last word.

[1] `https://github.com/espnet/espnet/tree/master/egs2/librispeech_100/asr1`

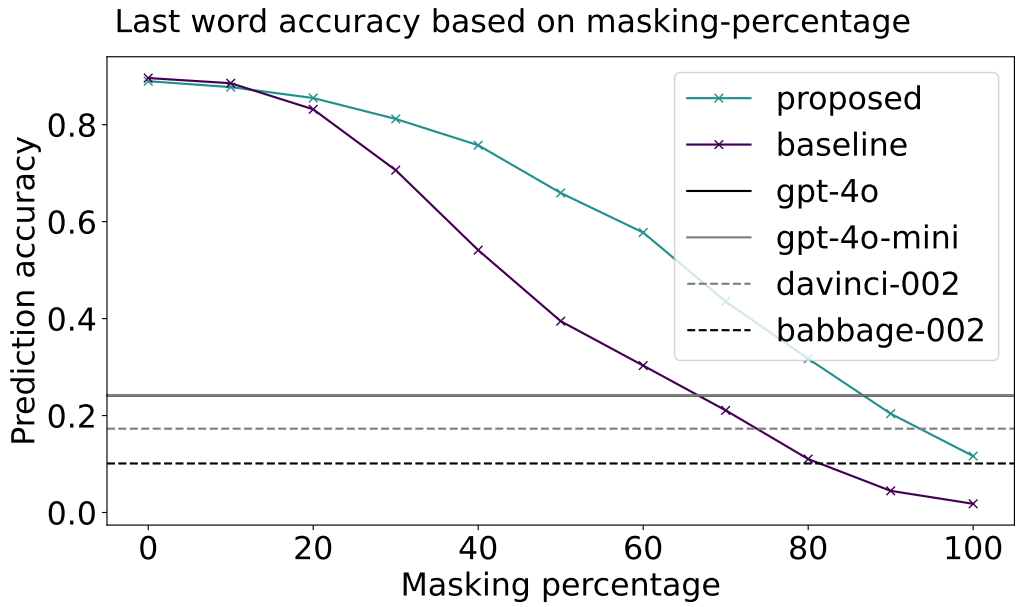


Figure 5.7.: Last word accuracy based on masking-percentage. The LMs `gpt-4o`, `gpt-4o-mini`, `babbage-002` and `davinci-002` are used as comparison for when no acoustic information is available at all. The proposed model does better than the LMs starting from about 20% available acoustic information.

Specifically, we run inference on the `test-clean` dataset of LS100 with $0, 10, \ldots, 90, 100\%$ of the audio of the last word masked for both the baseline and the proposed model. We do so using the MFA alignments, where the length of the last word is defined as

$$\Delta_{t,w_N} := w_{stop,w_N} - w_{start,w_N}.$$

Based on this, we mask different percentages of the last word, i.e., the interval

$$\left[w_{stop,w_N} - \frac{\Delta_{t,w_N}}{100} \cdot \aleph, t_{\text{EOA}} \right]$$

for a masking percentage of $\aleph = 10\%$. The prediction accuracies of the LMs `gpt-4o`, `gpt-4o-mini`, `babbage-002` and `davinci-002` are also shown as a comparison with no additional acoustic information. The models `gpt-4o` and `gpt-4o-mini` are instruction finetuned. Additionally to receiving the ground truth text, preceeding the last token, that is to predicted, these models are prompted, as described in section A.5. The non-instruction finetuned models, just receive the preceeding text as input.

The results are shown in Figure 5.7. We can see, that for 100% masking, i.e., no additional acoustic information, the compared LMs do better, even though they are not trained on the same domain (LS100). We observe, that the proposed model outperforms the LMs, starting with about 20% of the audio being available. We assume, this is due to the start of the audio significantly decreasing the search space of possible next words.

In the following, we show 5 cherry-picked examples of the LS100 `test-clean` dataset. We compare the baseline and proposed model, of which the the last word input is 50%

masked, to show how acoustic information is used. We also show the ground truth and the prediction of the LM `gpt-4o` to showcase strong LM capabilities without acoustic context.

```
Example 1 (UID 1089-134686-0027):
GT:       HE CLASPED HIS HANDS ON THE DESK AND SAID
Baseline: HE CLASPED HIS HANDS ON THE DESK AND SAID
Proposed: HE CLASPED HIS HANDS ON THE DESK AND SAID
GPT-4o:   HE CLASPED HIS HANDS ON THE DESK AND LEANED

Example 2 (UID 1089-134686-0028):
GT:       THE RETREAT WILL BEGIN ON WEDNESDAY AFTERNOON IN HONOUR OF SAINT FRANCIS
↪  XAVIER WHOSE FEAST DAY IS SATURDAY
Baseline: THE RETREAT WILL BEGIN ON WEDNESDAY AFTERNOON IN HONOUR OF SAINT FRANCIS
↪  ZEVIOR WHOSE FEAST DAY IS SATIR
Proposed: THE RETREAT WILL BEGIN ON WEDNESDAY AFTERNOON IN HONOR OF SAINT FRANCE'S
↪  SAVIOUR WHOSE FEAST DAY IS SATURING
GPT-4o:   THE RETREAT WILL BEGIN ON WEDNESDAY AFTERNOON IN HONOUR OF SAINT FRANCIS
↪  XAVIER WHOSE FEAST DAY IS DECEMBER

Example 3 (UID 1089-134691-0007):
GT:       SOON THE WHOLE BRIDGE WAS TREMBLING AND RESOUNDING
Baseline: SOON THE WHOLE BRIDGE WAS TREMBLING AND RESENT
Proposed: SOON THE WHOLE BRIDGE WAS TREMBLING AND RESOLVED
GPT-4o:   SOON THE WHOLE BRIDGE WAS TREMBLING AND SHAKING

Example 4 (UID 1188-133604-0017):
GT:       THAT A STYLE IS RESTRAINED OR SEVERE DOES NOT MEAN THAT IT IS ALSO
↪  ERRONEOUS
Baseline: THAT A STYLE IS RESTRAINED OR SEVERE DOES NOT MEAN THAT IT IS ALSO A RUN
Proposed: THAT A STYLE IS RESTRAINED OR SEVERE DOES NOT MEAN THAT IT IS ALSO
↪  ERRONEOUS
GPT-4o:   THAT A STYLE IS RESTRAINED OR SEVERE DOES NOT MEAN THAT IT IS ALSO
↪  UNINSPIRING

Example 5 (UID 1221-135766-0007):
GT:       HESTER PRYNNE NEVERTHELESS THE LOVING MOTHER OF THIS ONE CHILD RAN LITTLE
↪  RISK OF ERRING ON THE SIDE OF UNDUE SEVERITY
Baseline: HESTER PRINT NEVERTHELESS THE LOVING MOTHER OF THIS ONE CHILD RAN LITTLE
↪  RISK OF AIRING ON THE SIDE OF UNDUCEVER
Proposed: HESTER PRINCE NEVERTHELESS THE LOVING MOTHER OF THIS ONE CHILD RAN LITTLE
↪  RISK OF AIRING ON THE SIDE OF UNDOSE OF AIRY
GPT-4o:   HESTER PRYNNE NEVERTHELESS THE LOVING MOTHER OF THIS ONE CHILD RAN LITTLE
↪  RISK OF ERRING ON THE SIDE OF UNDUE SEVERITY
```

In examples 1, 2, 3 and 4 we can see that the LM `gpt-4o` without access to acoustic information, generates a semantically plausible last word, but missing the ground truth. In

example 5, we see that the greate LM capabilities of `gpt-4o` manage to predict the correct word, while the ASR based models predict wrongly.

We also observe, that the baseline model stops at the cut-off of the audio in all cases which leads to predicted words, that sound similar to the cutoff. In example 2, SATUR~~DAY~~ becomes *SATIR*. In example 4, ERRON~~OUS~~ becomes *A RUN* and in example 5, UNDUE SEVER~~ITY~~ becomes *UNDUECEVER*.

In contrast, the proposed model uses the partial information to predict complete words. However, due to the inferior LM capabilities, compared to `gpt-4o`, not all predictions are correct (example 4) we also see some errors (example 2, 3 & 5)

5.3. Discontinued Approaches

During this thesis, we explored two additional strategies, which we named *distillation loss* and *tuple loss*. Unfortunately, neither approach yielded stable training nor led to improvements in model performance.

Distillation Loss A primary concern in our cross-attention based EOU prediction was that the decoder might not sufficiently attend to the positional encoding-only frames appended to the encoder outputs. To address this problem, we proposed the *distillation loss* method involving two forward passes during training.

In the first forward pass, the input is presented to the model without masking. We then use the decoder cross-attention weights from this initial forward pass as the "teacher" attention, for the second masked forward pass where the last encoder frames are masked. To guide the model's attention in the masked pass, we apply a Kullback-Leibler Divergence (KL-Divergence) loss between the cross-attention distributions of the teacher and student passes.

The distillation loss, $\mathcal{L}_{\text{KL}}$, is added into the total loss function, defined as follows:

$$\mathcal{L} = w_{\text{ctc}} \cdot \mathcal{L}_{\text{ctc}} + (1 - w_{\text{ctc}}) \cdot \mathcal{L}_{\text{att}} + w_{\text{KL}} \cdot \mathcal{L}_{\text{KL}}$$

where w_{ctc} and w_{KL} are weighting parameters that control the relative contribution of the CTC loss, attention loss, and KL-Divergence loss, respectively.

We conducted experiments with varying weights w_{KL} and applied the KL-Divergence loss to either all multi-head attention heads or a single head. In all cases, training collapsed after several epochs. We hypothesize that forcing the cross-attention on essentially empty vectors may have contributed to this collapse.

In Figure 5.8, we illustrate the divergence observed during training with the distillation loss method. Initially, the gradient norm increases, indicating a gradual accumulation of instability during training. Around training step 22,000, the gradient norm becomes NaN, resulting in a complete collapse of the training.

The KL-Divergence loss follows a similar trend, rising consistently and peaking at approximately iteration 23,000. After this peak, the KL-Divergence loss begins to decline. Shortly after this peak, the instability results in the collapse of the training process.

These trends were consistently observed across all experimental configurations, regardless of whether the KL-Divergence loss was applied to all multi-head attention heads,

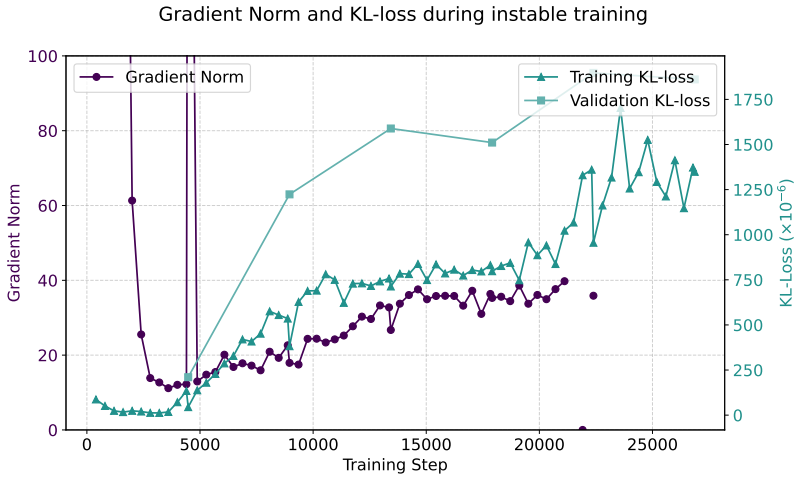


Figure 5.8.: Training divergence when using distillation loss after 5 epochs of training. The gradient norm gradually increases, until roughly at training step 25000 the training collapses. The KL-Divergence loss increases until peaking at about iteration 23000, after which it slowly decreases. Shortly after, the training collapses.

restricted to a single head, or limited to the final layer of the decoder's cross-attention mechanism.

Fortunately, further experimentation with our default masked training approach revealed that no additional incentive was needed for the cross-attention to focus on positional encoding-only frames. We thus abandoned this approach.

Tuple Loss An alternative approach considered for EOU prediction was predicting an additional scalar timing value t for each decoded token w. In this framework, the decoder would output a tuple (w, t) at each time step, where t represents the predicted timing in ms associated with the end of the utterance of token w.

To implement this, a separate linear layer was added on top of the final decoder hidden state, projecting it to a scalar timing prediction. The predicted timing t was then compared to the actual timing using a Mean Squared Error (MSE) loss, denoted as $\mathcal{L}_{\text{tuple}}$. The total loss function for training was thus defined as:

$$\mathcal{L} = w_{\text{ctc}} \cdot \mathcal{L}_{\text{ctc}} + (1 - w_{\text{ctc}}) \cdot \mathcal{L}_{\text{att}} + w_{\text{tuple}} \cdot \mathcal{L}_{\text{tuple}}$$

This approach aimed to provide a direct training signal for EOU prediction, potentially yielding more accurate predictions compared to the cross-attention-based method. However, experimental results demonstrated no significant improvement over the proposed cross-attention-based approach. Additionally, the cross-attention approach was more straightforward and did not require explicit word timing data for training, making it more practical.

6. Conclusion

In this thesis, we proposed an ASR model that simulates human anticipatory capabilities through the design of predictive EOU detection and predictive ASR tasks. We did so by using our proposed masked training approach, i.e., masking future segments of an utterance, thereby enabling the decoder to predict forthcoming words. Additionally, we proposed a cross-attention-based algorithm that leverages alignment information to accurately determine the timing of the EOU.

In section 1.1 we stated the research questions to answer in this thesis. This chapter attempts to answer these questions based on the experimental results discussed in chapter 5.

The experimental results showed that our model is capable of predicting upcoming words and estimating EOU timing up to 300 ms prior to the actual EOU.

Question 1: How well does EOU prediction with cross-attention work? The results show that, not only it is possible to predict the EOU with our cross-attention algorithm, but also our masked training approach showed improvements as well. Especially, for the less-challenging LS100 dataset, EOU prediction up to 300 ms before the actual EOU seems reasonably well. However, as in our Analysis we note, that this might be because of the additionally decoded words of our proposed model.

Question 2: How well can we predict future words? Similarly to Question 1, also up to 300 ms, FWER scores are reasonably well. Interestingly, FWER5 produces significantly better results. This could be explained, by synonyms and semantically grouped words that would all make sense in the context of the previously uttered sentence. In our Analysis, we also show that given 20% of a partially masked word our model starts to outperform off-the-self LMs with no additional acoustic information.

6.1. Discussion and Future Work

In our experiments, we demonstrated that our proposed model is capable of performing effectively when the EOU is known beforehand. Specifically, in our test scenarios, we masked a defined time span prior to the actual EOU, t_{EOU}. However, in real-world dialogue systems, the actual EOU is not known beforehand, and it remains uncertain whether the EOU will occur in the near future. This introduces a critical limitation: a supplementary module would be required to estimate whether t_{EOU} is imminent. Only under such conditions could our proposed model be used.

Additionally, our FWER@5 evaluation allows for up to five alternative ASR outputs. This raises the question of how such ASR outputs can be leveraged effectively by a subsequent response generation module. One potential approach might involve generating multiple responses corresponding to the top five ASR outputs. However, more sophisticated

methods could be developed to generate a semantically consistent response that accounts for synonyms or alternative phrasings across the ASR outputs.

Despite these limitations, this thesis serves as a foundational step toward predictive ASR and EOU detection. Substantial further research is required to evaluate the applicability of this approach within fully functional dialogue systems. This includes exploring the integration of predictive ASR with EOU estimation modules, and optimizing response generation strategies to handle the inherent uncertainty in ASR outputs.

Bibliography

[1] Dzmitry Bahdanau. “Neural machine translation by jointly learning to align and translate”. In: *arXiv preprint arXiv:1409.0473* (2014).

[2] William Chan et al. “Listen, attend and spell: A neural network for large vocabulary conversational speech recognition”. In: *Proc. ICASSP*. 2016, pp. 4960–4964.

[3] Shuo-Yiin Chang et al. “Low Latency Speech Recognition Using End-to-End Prefetching”. In: *Proc. Interspeech*. 2020, pp. 1962–1966.

[4] Xie Chen et al. “Developing real-time streaming transformer transducer for speech recognition on large-scale dataset”. In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 5904–5908.

[5] Jan K Chorowski et al. “Attention-based models for speech recognition”. In: *Proc. NeurIPS*. 2015, pp. 577–585.

[6] Zihang Dai. “Transformer-xl: Attentive language models beyond a fixed-length context”. In: *arXiv preprint arXiv:1901.02860* (2019).

[7] Linhao Dong, Feng Wang, and Bo Xu. “Self-attention aligner: A latency-control end-to-end model for asr using self-attention network and chunk-hopping”. In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 5656–5660.

[8] Erik Ekstedt and Gabriel Skantze. “TurnGPT: a Transformer-based Language Model for Predicting Turn-taking in Spoken Dialog”. In: *Findings of EMNLP*. 2020, pp. 2981–2990.

[9] Yifeng Fan et al. “Towards Accurate and Real-Time End-of-Speech Estimation”. In: *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2023, pp. 1–5.

[10] Jonas Gehring et al. “Convolutional sequence to sequence learning”. In: *International conference on machine learning*. PMLR. 2017, pp. 1243–1252.

[11] John J Godfrey, Edward C Holliman, and Jane McDaniel. “SWITCHBOARD: Telephone speech corpus for research and development”. In: *Proc. ICASSP*. 1992, pp. 517–520.

[12] Alex Graves. *Sequence Transduction with Recurrent Neural Networks*. 2012. arXiv: 1211.3711 [cs.NE]. URL: https://arxiv.org/abs/1211.3711.

[13] Alex Graves. “Sequence transduction with recurrent neural networks”. In: *Proc. ICML Representation Learning Workshop*. 2012.

[14] Anmol Gulati et al. “Conformer: Convolution-augmented transformer for speech recognition”. In: *arXiv preprint arXiv:2005.08100* (2020).

[15] Yanzhang He et al. “Streaming end-to-end speech recognition for mobile devices”. In: *Proc. ICASSP*. IEEE. 2019, pp. 6381–6385.

[16] Hirofumi Inaguma, Masato Mimura, and Tatsuya Kawahara. “CTC-synchronous training for monotonic attention model”. In: *arXiv preprint arXiv:2005.04712* (2020).

[17] Amirhossein Kazemnejad. “Transformer Architecture: The Positional Encoding”. In: *kazemnejad.com* (2019). URL: https://kazemnejad.com/blog/transformer_architecture_positional_encoding/.

[18] Ludwig Kürzinger et al. “Ctc-segmentation of large corpora for german end-to-end speech recognition”. In: *International Conference on Speech and Computer*. Springer. 2020, pp. 267–278.

[19] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[20] Stephen C Levinson and Francisco Torreira. “Timing in turn-taking and its implications for processing models of language”. In: *Frontiers in psychology* 6 (2015), p. 731.

[21] Bo Li et al. “A better and faster end-to-end model for streaming ASR”. In: *Proc. ICASSP*. 2021, pp. 5634–5638.

[22] Michael McAuliffe et al. “Montreal forced aligner: Trainable text-speech alignment using kaldi.” In: *Interspeech*. Vol. 2017. 2017, pp. 498–502.

[23] Niko Moritz, Takaaki Hori, and Jonathan Le. “Streaming automatic speech recognition with the transformer model”. In: *Proc. ICASSP*. 2020, pp. 6074–6078.

[24] Niko Moritz, Takaaki Hori, and Jonathan Le Roux. “Dual Causal/Non-Causal Self-Attention for Streaming End-to-End Speech Recognition”. In: *Proc. Interspeech*. 2017, pp. 1909–1913.

[25] Arun Narayanan et al. “Cascaded encoders for unifying streaming and non-streaming ASR”. In: *Proc. ICASSP*. 2021, pp. 5629–5633.

[26] Vassil Panayotov et al. “Librispeech: An ASR corpus based on public domain audio books”. In: *Proc. ICASSP*. 2015, pp. 5206–5210.

[27] Daniel S Park et al. “Specaugment: A simple data augmentation method for automatic speech recognition”. In: *arXiv preprint arXiv:1904.08779* (2019).

[28] Robert Porzel and Manja Baudis. “The tao of chi: Towards effective human-computer interaction”. In: *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*. 2004, pp. 209–216.

[29] Ofir Press, Noah A Smith, and Mike Lewis. “Train short, test long: Attention with linear biases enables input length extrapolation”. In: *arXiv preprint arXiv:2108.12409* (2021).

[30] Jin Sakuma, Shinya Fujie, and Tetsunori Kobayashi. “Response timing estimation for spoken dialog systems based on syntactic completeness prediction”. In: *Proc. SLT*. 2023, pp. 369–374.

[31] Matt Shannon et al. “Improved End-of-Query Detection for Streaming Speech Recognition”. In: *Proc. Interspeech*. 2017, pp. 1909–1913.

[32] Yangyang Shi et al. “Emformer: Efficient memory transformer based acoustic model for low latency streaming speech recognition”. In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 6783–6787.

[33] Koustuv Sinha et al. “The curious case of absolute position embeddings”. In: *arXiv preprint arXiv:2210.12574* (2022).

[34] Xingchen Song et al. “Trimtail: Low-latency streaming asr with simple but effective spectrogram-level length penalty”. In: *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2023, pp. 1–5.

[35] Andreas Stolcke et al. “Adaptive endpointing with deep contextual multi-armed bandits”. In: *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2023, pp. 1–5.

[36] Grant Strimel et al. “Lookahead when it matters: Adaptive non-causal transformers for streaming neural transducers”. In: *Proc. ICML*. 2023, pp. 32654–32676.

[37] Emiru Tsunoo, Yosuke Kashiwagi, and Shinji Watanabe. “Streaming transformer ASR with blockwise synchronous beam search”. In: *Proc. SLT*. 2021, pp. 22–29.

[38] Emiru Tsunoo et al. “Towards online end-to-end transformer automatic speech recognition”. In: *arXiv preprint arXiv:1910.11871* (2019).

[39] Ashish Vaswani et al. “Attention is all you need”. In: *Proc. NeurIPS*. 2017, pp. 5998–6008.

[40] Alexander Waibel et al. “Phoneme recognition using time-delay neural networks”. In: *Backpropagation*. Psychology Press, 2013, pp. 35–61.

[41] Yu-An Wang and Yun-Nung Chen. “What do position embeddings learn? an empirical study of pre-trained language model positional encoding”. In: *arXiv preprint arXiv:2010.04903* (2020).

[42] Shinji Watanabe et al. “ESPnet: End-to-end speech processing toolkit”. In: *arXiv preprint arXiv:1804.00015* (2018).

[43] Shinji Watanabe et al. “Hybrid CTC/attention architecture for end-to-end speech recognition”. In: *IEEE Journal of Selected Topics in Signal Processing* 11.8 (2017), pp. 1240–1253.

[44] Jiahui Yu et al. “Dual-mode ASR: Unify and improve streaming ASR with full-context modeling”. In: *Proc. ICLR*. 2021.

[45] Jiahui Yu et al. “Fastemit: Low-latency streaming asr with sequence-level emission regularization”. In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 6004–6008.

[46] Bolaji Yusuf et al. “Speculative Speech Recognition by Audio-Prefixed Low-Rank Adaptation of Language Models”. In: *Proc. Interspeech*. 2024, pp. 792–796.

[47] Qian Zhang et al. “Transformer transducer: A streamable speech recognition model with transformer encoders and RNN-T loss”. In: *Proc. ICASSP*. 2020, pp. 7829–7833.

[48] Qian Zhang et al. “Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss”. In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 7829–7833.

[49] Huaibo Zhao et al. “Conversation-oriented ASR with multi-look-ahead CBS architecture”. In: *Proc. ICASSP*. 2023, pp. 1–5.

[50] Huaibo Zhao et al. “Mask-CTC-based Encoder Pre-training for Streaming End-to-End Speech Recognition”. In: *Proc. EUSIPCO*. 2023, pp. 56–60.

A. Appendix

A.1. ESPnet recipes

The ESPnet recipes are `yaml` files that define the training settings. For clarity, we split the configuration into two parts: `upstream`, which contains the default ESPnet settings, and `custom`, which includes the options introduced specifically for this thesis.

For both `LS100` and `SWBD` we use the same settings except for the setting `warmup_steps` and `max_epoch`.

For `warmup_steps`, we set to 15000 for `LS100` and 25000 for `SWBD`. For `max_epoch`, we set to 70 for `LS100` and 100 for `SWBD`.

For the custom setting the following properties were introduced: `blocks_training` and `random_blocks` are the heart of our masked training approach. `blocks_training` essentially defines how much audio should be masked during training. Each "block" defines a single frame before the audio is fed through the convolutional subsampling of the encoder (section 2.2). Thus, each "block" equals to 10ms of audio. Specifically, we sample $x \in \{0, 1, \ldots, \texttt{blocks\_training}\}$ and mask x frames before t_{EOU}, as detailed in subsection 4.2.2.

Analogously, `random_blocks` are also defined to be in 10ms intervals. Also detailed in subsection 4.2.2, these "blocks" are positional encoding only and serve as a variability in input length because of the variation in silence duration of the datasets. We also sample uniformly out of $\xi \in \{-\texttt{random\_blocks}, \ldots, -1, 0, 1, \ldots, \texttt{random\_blocks}\}$ and use the sampled xi to add or subtract positional-encoding-only frames to the input.

Even though the options `uniform_sampling` and `is_causal` exist, these options are always set to `true` in our experiments. This is because we always want to use a causal encoder to mimic a dialogue system and we found uniform sampling lead to better results than Gaussian sampling.

The options `is_self_distilling`, `use_last_head_distill` and `use_last_layer_distill` are all used for the distillation loss, `is_self_distilling` enables or disables the loss, `use_last_head_distill` and `use_last_layer_distill` define, on which attention heads and layers the loss is applied to. The weight for the distillation loss is set in the upstream `model_conf: distill_weight`.

Similarly, the options `use_timing_loss` is used to disable or enable the timing loss. The options `use_single_head` and `only_last_layer_timing` are analogous to `use_last_head_distill` and `use_last_layer_distill`. `only_last_timing` further specifies to only apply the loss the the last decoded token — the `<EOS>` token. `timing_loss_weight` sets the weight for the timing loss.

custom

```
blocks_training: 50                 # 1 block = 10ms
random_blocks: 20                   # +- blocks for randomness
uniform_sampling: true

is_causal: true

# Distillation loss
is_self_distilling: false
use_last_head_distill: false
use_last_layer_distill: false

# Timing loss
use_timing_loss: false
only_last_timing: false
use_single_head: false
only_last_layer_timing: false
timing_loss_weight: 0
```

upstream

```
encoder: conformer
encoder_conf:
    output_size: 256
    attention_heads: 4
    linear_units: 1024
    num_blocks: 12
    dropout_rate: 0.1
    positional_dropout_rate: 0.1
    attention_dropout_rate: 0.1
    input_layer: conv2d
    normalize_before: true
    macaron_style: true
    rel_pos_type: latest
    pos_enc_layer_type: abs_pos
    selfattention_layer_type: selfattn
    activation_type: swish
    use_cnn_module: true
    cnn_module_kernel: 31

decoder: transformer
decoder_conf:
    attention_heads: 4
    linear_units: 2048
    num_blocks: 6
```

```
    dropout_rate: 0.1
    positional_dropout_rate: 0.1
    self_attention_dropout_rate: 0.1
    src_attention_dropout_rate: 0.1

model_conf:
    ctc_weight: 0.3
    distill_weight: 0
    lsm_weight: 0.1
    length_normalized_loss: false

ctc_conf:
    ignore_nan_grad: true

frontend_conf:
    n_fft: 512
    win_length: 400
    hop_length: 160

seed: 2022
log_interval: 400
num_att_plot: 0
num_workers: 4
sort_in_batch: descending       # how to sort data in making batch
sort_batch: descending          # how to sort created batches
batch_type: numel
batch_bins: 16000000
accum_grad: 4
max_epoch: 70
patience: none
init: none
best_model_criterion:
-   - valid
    - acc
    - max
keep_nbest_models: 10

use_amp: false
cudnn_deterministic: false
cudnn_benchmark: false

optim: adam
optim_conf:
    lr: 0.002
    weight_decay: 0.000001
```

```
scheduler: warmuplr
scheduler_conf:
# Librispeech: 15.000, SWBD: 25.000
    warmup_steps: 15000

specaug: specaug
specaug_conf:
    apply_time_warp: true
    time_warp_window: 5
    time_warp_mode: bicubic
    apply_freq_mask: true
    freq_mask_width_range:
    - 0
    - 27
    num_freq_mask: 2
    apply_time_mask: true
    time_mask_width_ratio_range:
    - 0.
    - 0.05
    num_time_mask: 5
```

A.2. Model Summary

```
ESPnetASRModel(
  (frontend): DefaultFrontend(
    (stft): Stft(n_fft=512, win_length=400, hop_length=160, center=True,
    ↪  normalized=False, onesided=True)
    (frontend): Frontend()
    (logmel): LogMel(sr=16000, n_fft=512, n_mels=80, fmin=0, fmax=8000.0,
    ↪  htk=False)
  )
  (specaug): SpecAug(
    (time_warp): TimeWarp(window=5, mode=bicubic)
    (freq_mask): MaskAlongAxis(mask_width_range=[0, 27], num_mask=2, axis=freq)
    (time_mask): MaskAlongAxisVariableMaxWidth(mask_width_ratio_range=[0.0, 0.05],
    ↪  num_mask=5, axis=time)
  )
  (normalize):
  ↪  GlobalMVN(stats_file=exp/asr_stats_raw_en_bpe5000_sp/train/feats_stats.npz,
  ↪  norm_means=True, norm_vars=True)
  (encoder): ConformerEncoder(
    (embed): Conv2dSubsampling(
```

```
    (conv): Sequential(
      (0): Conv2d(1, 256, kernel_size=(3, 3), stride=(2, 2))
      (1): ReLU()
      (2): Conv2d(256, 256, kernel_size=(3, 3), stride=(2, 2))
      (3): ReLU()
    )
    (out): Sequential(
      (0): Linear(in_features=4864, out_features=256, bias=True)
      (1): PositionalEncoding(
        (dropout): Dropout(p=0.1, inplace=False)
      )
    )
  )
  (encoders): MultiSequential(
    (N): EncoderLayer (repeated N times)
      - MultiHeadedAttention
      - PositionwiseFeedForward
      - FeedForwardMacaron
      - ConvolutionModule
      - LayerNorm
      - Dropout
  )
)
(decoder): TransformerDecoder(
  (embed): Sequential(
    (0): Embedding(5000, 256)
    (1): PositionalEncoding(
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
  (decoders): MultiSequential(
    (M): DecoderLayer (repeated M times)
      - SelfAttention
      - SourceAttention
      - PositionwiseFeedForward
      - LayerNorm
      - Dropout
  )
  (output_layer): Linear(in_features=256, out_features=5000, bias=True)
)
(criterion_att): LabelSmoothingLoss(
  (criterion): KLDivLoss()
```

```
  )
  (ctc): CTC(
    (ctc_lo): Linear(in_features=256, out_features=5000, bias=True)
    (ctc_loss): CTCLoss()
  )
)

Model Summary:
- Class Name: ESPnetASRModel
- Total Number of model parameters: 33.44 M
- Number of trainable parameters: 33.44 M (100.0%)
- Size: 133.75 MB
- Type: torch.float32

Optimizer:
Adam (
  - amsgrad: False
  - betas: (0.9, 0.999)
  - eps: 1e-08
  - lr: 1.3333333333333336e-07
  - weight_decay: 1e-06
)
```

A.3. SWBD example conversation

The following shows an example conversation of the SWBD dataset.

```
sw02010-B_000000-000192.txt this is archie
sw02010-A_000179-000356.txt hi archie i'm sharon
sw02010-B_000308-000539.txt have you seen dancing with wolf
sw02010-A_000490-001303.txt yeah i've seen that that's uh that was a really good
↪  movie probably one of the best things about it was the scenery and uh
sw02010-A_001303-001756.txt i thought the story was pretty good too i th- i think
↪  kevin costner did a really good job with it
sw02010-B_001676-001961.txt have you ever lived in that part of the country
sw02010-A_001856-002057.txt no i haven't
sw02010-B_001961-002156.txt have you ever visited it
sw02010-A_002057-002726.txt um i've visited the wyoming area i'm not sure exactly
↪  where dances with wolves was filmed
sw02010-B_002653-003045.txt i think it was the black hills of south dakota
sw02010-A_002947-003284.txt could be i- i haven't been to south dakota
sw02010-A_003284-003530.txt ha- have you been up to that
```

```
sw02010-B_003407-004004.txt well i lived in omaha for five winters and that
↪  rolling kind of uh
sw02010-A_003530-003827.txt oh okay
sw02010-A_003931-004091.txt terrain
sw02010-B_004004-004695.txt yeah is is fairly fami- the thing that i thought was
↪  interesting was that the critics apparently it's gonna win everything
sw02010-A_004611-004750.txt really
sw02010-B_004695-005293.txt uh and i had been told you know you wouldn't notice
↪  that it was three hours long and all this kind of
sw02010-A_005106-006570.txt that's true i agree with that um i- i noticed
↪  yesterday in the paper something said that it i think it's been nominated for
↪  twelve awards and and all the critics initially said that you know it
↪  wouldn't go anywhere that it was just going to be a dud so it it has
↪  surprised everybody
sw02010-B_006442-007507.txt well the interesting thing was is i had heard that
↪  and i- i- i tend to i think overreact occasionally when somebody tells me
↪  it's that great and and it was the thing is it was a it was a good story
sw02010-A_007090-007231.txt uh-huh
sw02010-A_007441-007559.txt right
sw02010-B_007507-008297.txt and and and i guess that's what i really like
↪  although i must admit i did look at my watch after about an hour yeah
sw02010-A_008122-008248.txt did you
sw02010-A_008248-008509.txt have you seen sleeping with the enemy
sw02010-B_008404-008819.txt no i've heard i've heard that's really great though
sw02010-A_008721-009174.txt y- you have to go see that one and how about silence
↪  of the lambs go
sw02010-B_008977-009264.txt no you must really keep up
sw02010-A_009174-009951.txt i do i go every weekend i- i uh those are two
↪  definite must see movies i think
sw02010-B_009838-010567.txt we- isn't isn't sleeping with the enemy isn't that a
↪  i- is it is it a terror movie or is it just suspenseful
sw02010-A_010483-011311.txt it's suspenseful i don't think it's very t- i mean
↪  there's not really any uh blood and guts in it or anything like that it's
↪  it's more suspense
sw02010-B_012460-012985.txt do you do you listen to gary cohill cohi- is uh you
↪  know who he is
sw02010-A_012665-012827.txt no huh-uh
sw02010-A_012933-013293.txt yeah somebody in south carolina told me about him
sw02010-B_013178-013797.txt he's a the movie critic um oh you've oh okay he's a
↪  movie critic -n channel eight in dallas
sw02010-A_013716-013848.txt yeah
sw02010-B_013797-014286.txt and he does uh he has a talk show on the
sw02010-B_014286-015495.txt k l i f anyway it's on from seven to nine or
↪  something and and and people call in and it's if you keep up with movies it's
↪  kind of interesting there's a certain amount of drivel that they do they've
↪  got uh they've got a couple of kids
sw02010-B_015495-015882.txt ten or twelve years old and they call in and they
↪  review movies but it's uh
```

```
sw02010-A_015771-015905.txt oh
sw02010-B_015882-016763.txt he anyway he it's interesting you listen to him and
↪  then you you go watch the movie fact they had people had just seen i was
↪  listening sunday night a little bit when i was uh going to
sw02010-A_016540-016659.txt uh-huh
sw02010-B_016763-017892.txt pick up my daughter but anyway it was uh i- the it
↪  it's interesting though the the the difficulty with with dancing with wolves
↪  is that when you make a movie like that and you produce it and then you star
↪  in it
sw02010-B_017892-018425.txt uh the question is did he did he really know it was
↪  gonna be good or did he just do it
sw02010-A_018351-019289.txt i think that i think he really his heart was in it
↪  but i d- i don't think he really knew it was gonna be as big as it was i
↪  think it was something that he really wanted to do
sw02010-A_019289-020140.txt he wanted to direct it he wanted to a- to star in it
↪  you know he he enjoyed the story line and i think he just really w- he really
↪  wanted it whether it
sw02010-A_020140-020758.txt whether it won all kinds of awards or whether it just
↪  was okay at the box office i think he would have been happy because
sw02010-A_020758-021619.txt i think that i think he did a good job and and the
↪  self-satisfaction he got out of it is much greater than any awards that they
↪  can give him
sw02010-B_021561-022101.txt do you know who the guy was that was playing the uh
↪  the the wagon driver
sw02010-A_022095-022267.txt um
sw02010-B_022101-022607.txt a little piece of trivia you know the guy when he
↪  first headed out from the army post
sw02010-A_022576-022931.txt yeah who no i don't know who that guy is
sw02010-B_022805-023123.txt he he plays on murphy brown
sw02010-A_023043-023215.txt oh he does
sw02010-B_023123-023351.txt yes he's
sw02010-A_023215-023524.txt as a recurring character every week
sw02010-B_023351-023668.txt he's eldon her housepainter
sw02010-B_023668-023887.txt do you believe that
sw02010-A_024382-024536.txt huh-uh
sw02010-A_024955-025096.txt yeah
sw02010-B_025022-025312.txt you know with the beard and all that stuff i mean
↪  it's uh
sw02010-B_025312-026268.txt yeah real scruffy looking and it wa- it was really
↪  funny it's uh that that he winds up playing in the movie but i thought it's
↪  good that you know it was a lot of fun
sw02010-A_026199-026375.txt yeah
sw02010-B_026268-026809.txt i don't know how long this conversation is supposed
↪  to go but we're at about five minutes i should think we've done enough
sw02010-A_026720-027127.txt you think so i mean i haven't been watching my watch
↪  um
sw02010-B_027434-027803.txt well i don't know wh- how do we end this thing i
↪  think it just says hang up
```

A.4. LS100 example

The following is an example segment of the LS100 `dev-clean` dataset.

```
1272-128104-0000 MISTER QUILTER IS THE APOSTLE OF THE MIDDLE CLASSES AND WE ARE
↪  GLAD TO WELCOME HIS GOSPEL
1272-128104-0001 NOR IS MISTER QUILTER'S MANNER LESS INTERESTING THAN HIS MATTER
1272-128104-0002 HE TELLS US THAT AT THIS FESTIVE SEASON OF THE YEAR WITH
↪  CHRISTMAS AND ROAST BEEF LOOMING BEFORE US SIMILES DRAWN FROM EATING AND ITS
↪  RESULTS OCCUR MOST READILY TO THE MIND
1272-128104-0003 HE HAS GRAVE DOUBTS WHETHER SIR FREDERICK LEIGHTON'S WORK IS
↪  REALLY GREEK AFTER ALL AND CAN DISCOVER IN IT BUT LITTLE OF ROCKY ITHACA
1272-128104-0004 LINNELL'S PICTURES ARE A SORT OF UP GUARDS AND AT EM PAINTINGS
↪  AND MASON'S EXQUISITE IDYLLS ARE AS NATIONAL AS A JINGO POEM MISTER BIRKET
↪  FOSTER'S LANDSCAPES SMILE AT ONE MUCH IN THE SAME WAY THAT MISTER CARKER USED
↪  TO FLASH HIS TEETH AND MISTER JOHN COLLIER GIVES HIS SITTER A CHEERFUL SLAP
↪  ON THE BACK BEFORE HE SAYS LIKE A SHAMPOOER IN A TURKISH BATH NEXT MAN
1272-128104-0005 IT IS OBVIOUSLY UNNECESSARY FOR US TO POINT OUT HOW LUMINOUS
↪  THESE CRITICISMS ARE HOW DELICATE IN EXPRESSION
1272-128104-0006 ON THE GENERAL PRINCIPLES OF ART MISTER QUILTER WRITES WITH
↪  EQUAL LUCIDITY
1272-128104-0007 PAINTING HE TELLS US IS OF A DIFFERENT QUALITY TO MATHEMATICS
↪  AND FINISH IN ART IS ADDING MORE FACT
1272-128104-0008 AS FOR ETCHINGS THEY ARE OF TWO KINDS BRITISH AND FOREIGN
1272-128104-0009 HE LAMENTS MOST BITTERLY THE DIVORCE THAT HAS BEEN MADE BETWEEN
↪  DECORATIVE ART AND WHAT WE USUALLY CALL PICTURES MAKES THE CUSTOMARY APPEAL
↪  TO THE LAST JUDGMENT AND REMINDS US THAT IN THE GREAT DAYS OF ART MICHAEL
↪  ANGELO WAS THE FURNISHING UPHOLSTERER
1272-128104-0010 NEAR THE FIRE AND THE ORNAMENTS FRED BROUGHT HOME FROM INDIA ON
↪  THE MANTEL BOARD
1272-128104-0011 IN FACT HE IS QUITE SEVERE ON MISTER RUSKIN FOR NOT RECOGNISING
↪  THAT A PICTURE SHOULD DENOTE THE FRAILTY OF MAN AND REMARKS WITH PLEASING
↪  COURTESY AND FELICITOUS GRACE THAT MANY PHASES OF FEELING
1272-128104-0012 ONLY UNFORTUNATELY HIS OWN WORK NEVER DOES GET GOOD
1272-128104-0013 MISTER QUILTER HAS MISSED HIS CHANCE FOR HE HAS FAILED EVEN TO
↪  MAKE HIMSELF THE TUPPER OF PAINTING
1272-128104-0014 BY HARRY QUILTER M A
```

A.5. OpenAI 4o 4o-mini prompt

For our last word prediction experiments, we use the following prompt/python snippet, with `line` being a line of the test with the last word truncated:

```python
# Perform GPT-4o-mini completion for each line
completion = client.chat.completions.create(
```

```
    # or model="gpt-4o"
    model="gpt-4o-mini",
    messages=[
        {"role": "system", "content": "You are given an uncompleted sentence. The last word
        ↪  is missing. Fill the last word."},
        {"role": "user", "content": line}
    ]
)
```