The Role of Future-Context Prediction in Low-Latency Speech Translation

Bachelor's Thesis of

Emir Ünlü

Artificial Intelligence for Language Technologies (AI4LT) Lab
Institut for Anthropomatics and Robotics (IAR)
KIT Department of Informatics

Reviewer:	Prof. Dr. Jan Niehues
Second reviewer:	Prof. Dr.-Ing. Rainer Stiefelhagen
Advisor:	M.Sc. Danni Liu
Second advisor:	M.Sc. Sai Koneru

25. February 2025 – 25. June 2025

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.

Karlsruhe, DATE

. . . . . .

(Emir Ünlü)

Abstract

Low-latency speech translation aims to deliver real-time translations of spoken language with minimal delay and high accuracy. However, early translation often suffers from limited context, leading to degraded performance. A naive way to overcome this issue is to make the system wait for enough input tokens, which in turn increases the latency. This thesis investigates an alternative solution, the use of unobserved future-context prediction, to enhance both latency and translation quality in cascaded speech translation systems.

By integrating lightweight large language models (LLMs) such as Qwen2.5-0.5B into a cascaded low-latency speech translation system to predict the next tokens in the so far predicted transcription/translation, the system gains additional context at each segment step. Moreover, several methods, such as using previous context, filtering future-context predictions based on a prediction threshold, are introduced to improve the accuracy of unobserved future-context prediction.

A comprehensive experimental setup evaluates step by step the performance of a cascaded speech translation system, including offline/low-latency performance of isolated ASR/MT and cascaded speech translation. In the next step, the setup examines the impact of different hyperparameters, such as input size, previous context size, prediction threshold, and timing of prediction, on the accuracy of future-context prediction. At the end, future-context-prediction-integrated low latency speech translation system is evaluated with many different scenarios. The results on the English-German subset of the Europarl-ST dataset show that incorporating selected future-context predictions is a promising method to improve system's latency with a negligible impact on translation quality.

Zusammenfassung

Die sprachliche Übersetzung mit niedriger Latenz zielt darauf ab, gesprochene Sprache in Echtzeit mit minimaler Verzögerung und hoher Genauigkeit zu übersetzen. Frühzeitige Übersetzungen leiden jedoch häufig unter begrenztem Kontext, was zu einer verringerten Leistungsfähigkeit führt. Eine naive Lösung für dieses Problem besteht darin, das System warten zu lassen, bis genügend Eingangstoken verfügbar sind – was jedoch die Latenz erhöht. Diese Arbeit untersucht eine alternative Lösung: die Vorhersage von nicht beobachtetem zukünftigen Kontext, um sowohl die Latenz als auch die Übersetzungsqualität in kaskadierten Sprachübersetzungssystemen zu verbessern.

Durch die Integration leichtgewichtiger großer Sprachmodelle (LLMs) wie Qwen2.5-0.5B in ein kaskadiertes System zur Sprachübersetzung mit niedriger Latenz, welches die nächsten Token in der bisher vorhergesagten Transkription/Übersetzung vorhersagt, erhält das System bei jedem Segmentierungsschritt zusätzlichen Kontext. Darüber hinaus werden verschiedene Methoden eingeführt – wie die Nutzung von vorherigem Kontext und das Filtern von Vorhersagen des zukünftigen Kontexts basierend auf einem Vorhersageschwellenwert –, um die Genauigkeit der Vorhersage von nicht beobachtetem zukünftigen Kontext zu verbessern.

Ein umfassendes experimentelles Setup bewertet schrittweise die Leistung eines kaskadierten Sprachübersetzungssystems, einschließlich der Offline- bzw. niedrigen Latenzleistung von isolierter ASR/MT sowie der kaskadierten Sprachübersetzung. Im nächsten Schritt untersucht das Setup den Einfluss verschiedener Hyperparameter – wie Eingabegröße, Größe des vorherigen Kontexts, Vorhersageschwellenwert und Zeitpunkt der Vorhersage – auf die Genauigkeit von Vorhersage des zukünftigen Kontexts. Abschließend wird das mit Vorhersage des zukünftigen Kontexts integrierte System zur Sprachübersetzung mit niedriger Latenz unter verschiedenen Szenarien evaluiert. Die Ergebnisse auf dem Englisch-Deutsch-Teil des Europarl-ST-Datensatzes zeigen, dass die Verwendung von ausgewählten Vorhersagen des zukünftigen Kontexts eine vielversprechende Methode zur Verbesserung der Systemlatenz mit vernachlässigbarem Einfluss auf die Übersetzungsqualität ist.

Contents

List of Figures

1. Introduction

1.1. Motivation

Translation is becoming increasingly crucial with the accelerating globalization. The number of international summits, conferences, and organizations has increased drastically in the last few decades. In addition to this, more people have access to world-wide live broadcasts thanks to digital platforms. However, it is usually the case that the speech is delivered in a language that the audience is not familiar with. All of these developments indicate that low-latency, aka real-time speech translation, is the key to modern communication.

In low-latency speech translation, the goal is to translate the spoken utterance into the target language with as little delay as possible. Unlike pre-recorded or consecutive translation, the interpreter does not wait for the speaker to complete their sentence or entire speech.

Recent developments in speech processing and natural language processing (NLP) disciplines, such as automatic speech recognition (ASR), neural machine translation (neural MT), and speech synthesis have enabled us to implement low-latency speech translation with AI-driven models.

However, the quality and latency of the early AI-driven low-latency speech translation systems have been far from being sufficient for practical use. To address this, several methods have been developed to enhance the performance of the automatic low-latency speech translation, such as partial hypothesis selection, training models with partial sentence translations, etc.

Currently, the focus is on leveraging unobserved future-context prediction in low-latency translation to further enhance performance. Unobserved future-context prediction involves predicting unobserved parts of the spoken utterance, allowing the translation to begin earlier. By providing the model with unobserved future-context as an additional context during the early stages of translation, this approach can improve the latency and accuracy of the translation. However, research on this topic remains limited. Therefore, my thesis will analyze whether unobserved future-context prediction can improve the latency and/or accuracy of low-latency speech translation models.

1.2. Research Questions

In this work, the main objective is to explore the implementation techniques and impacts of unobserved future-context prediction in the field of low-latency speech translation. The thesis addresses the following research questions:

- **Research Question 1: How can unobserved future-context be predicted?**

  Future-context prediction is not a trivial task. Therefore, the following points must be clarified: algorithms/models to be used, the scope of prediction (whether to predict a few tokens or complete sentence), and amount of input context required.

- **Research Question 2: How can unobserved future-context prediction be integrated into existing low-latency speech translation systems?**

  There are essentially two ways to integrate future-context prediction: predicting future-context at the end of each ASR stage or at the end of each MT stage. What is more, the generated future-context prediction must be verified before being committed. Providing the system with incorrect context may negatively affect translation quality. In this work, different approaches are tested to determine which ones perform better.

- **Research Question 3: How do different configurations of the future-context prediction module impact low-latency speech translation performance?**

  One of the drawbacks of low-latency speech translation is that the model must wait to receive sufficient context before starting the translation. Otherwise, the translation quality drops significantly, as explained in Section 5. By using future-context prediction, the model is provided with additional context and can therefore start the translation earlier. This may result in lower latency. Moreover, there is a possibility that the translation accuracy improves thanks to having a richer context in the early stages of translation.

2. Background and Related Work

Before examining the actual work, it is important to review basic information and related studies on the topic. This section covers the definition of low-latency speech translation and the implementation logic behind it. In addition, methods developed to improve the performance of such systems are also presented.

2.1. System Architecture

Essentially, there are two architectures to implement speech translation systems:

- **End-to-End (E2E) Architecture:** by using a single model, this architecture enables the translation of spoken utterances without relying on intermediate text representations. Thanks to its simple structure, the computational overhead and the inference latency are reduced. However, it is less efficient due to the integration of ASR and MT tasks into a single model [8].

- **Cascaded Architecture:** consists of two different components that handle ASR and MT tasks separately. The ASR model is trained to transcribe spoken utterances into corresponding textual representations. In the next step, the MT model receives the transcribed text in the source language as input and generates its translation in the target language. Thanks to its modular structure, each module can be specialized for its specific task, which enhances the overall performance significantly. Furthermore, each component of the system can be developed, tested and replaced independently, making the entire system easier to manage [8], [2].

2.2. Low-Latency Speech Translation

2.2.1. Task Definition

In this work, speech translation is defined as the process of translating a spoken utterance in a source language into its textual representation in a target language. And low-latency speech translation refers to speech translation performed in real-time. The goal is to reduce the delay between the spoken utterance and the translation while maintaining the translation accuracy. It is also known as simultaneous or real-time speech translation.

There are different ways to implement an AI-driven low-latency speech translation system. In the cascaded approach, the following two tasks are performed by different models, one for each task [8]:

- **Automatic Speech Recognition:** automatic process of converting speech signals to corresponding text [15].

- **Machine Translation**: the study of how to use computers to translate texts from one natural language to another. State-of-the-art approach for machine translation is the neural machine translation which is based on deep learning techniques [23].

In the E2E approach, these two tasks are merged into a single task and performed by a single E2E model. The E2E model does not rely on intermediate text representations, as it processes input speech directly into the translated text [8].

2.2.2. Incremental Decoding

[6] introduces a method called **incremental decoding** which is used to implement low-latency speech translation. It involves dividing the spoken utterance into fixed-size chunks and feeding them incrementally as inputs to the encoder-decoder-based AI model. As the model receives a new chunk, all segments received so far are decoded. Moreover, the decoder is conditioned on the output of the previous chunk step. In doing so, an attention mechanism is established that spans over all chunk steps processed so far, rather than focusing on the current chunk step alone. The non-committed portion of the output which is generated at the end of the chunk step is committed and no longer modified to prevent the translation from being overwritten at each chunk step.

There are two methods to condition the decoding process with the output of the previous chunk step: using buffered decoder state or forced decoding of the output tokens from the previous chunk step.

2.2.3. Input Segmentation Strategies

Simultaneous speech translation requires the segmentation of the input speech into smaller units. Segmented input is then provided to the system incrementally. There are three main strategies to achieve segmentation:

- **Fixed-length Segmentation:** divides the speech into equally-sized input segments with a fixed frame length [7].

- **Word-based Segmentation:** uses detectors to locate word boundaries of the input and divides input into segments accordingly [7].

- **Adaptive Segmentation:** reads the acoustic information from the encoder and locates the boundaries of meaningful speech units [7].

Fixed-length segmentation offers a solution easier to implement, but it lacks the ability to locate semantic boundaries of the input speech. It is an important ability, since poor acoustic information in the input may lead to the generation of poor-quality outputs.

2.2.4. Simultaneous Read-Write Policies

To implement low-latency incremental speech translation, numerous policies are developed. They can be categorized into fixed and flexible policies [7]. Fixed policies are easier to implement. On the other hand, flexible policies can determine dynamically when to read more input and write an output by analyzing the current partial input and output.

Fixed policies are listed below:

- **Hold-n:** removes the last n tokens from the output of current chunk step [6].
- **Wait-k:** waits until the model receives the first k input segments and output at a fixed rate [9].

Besides that, there are different versions of the wait-k strategy, such as wait-k-stride-n [27].

And flexible policies are:

- **Continuous Integrate-and-Fire (CIF):** a monothonic alignment method which uses a weight prediction network. If the accumulated weights fall below a predefined threshold value, it proceeds to the next encoder step. Otherwise, a operation called **integrate and fire** is triggered to retain the remaining weight for the next integration. Then the CIF generates an integrated embedding which is sent to the decoder. This step is called firing [1].
- **Cross Attention Augmented Transducer (CAAT):** based on the Transducer framework, a model is trained to optimize the decision-making of the read-write policies by calculating a latency loss in addition to the accuracy loss [5].
- **Encoder-Decoder Attention (EDATT):** leverages the attention matrix of the encoder-decoder model. The received partial input is analyzed in terms of its calculated attention. If the attention is concentrated on the most recent tokens, then the model decides to read more input segments. Otherwise it emits a partial translation, as it thinks that it received enough input context [17].
- **Incremental Blockwise Beam-Search (IBWBS)** a policy which dynamically manages generated beams. Beams with a unreliable hypothesis are considered problematic and therefore halted. Meanwhile, other beams are allowed to continue [19].
- **Local Agreement:** take the longest common prefix of the current chunk step's output and the-not-yet committed output of the previous chunk step [6].
- **Monotonic Multi-head Attention (MMA):** utilizes multi-head attention to realize flexible decision-making. Each head within a layer calculates an independent step probability which determines when to read and write during the simultaneous translation [10].

2.3. Methods for Improving Performance of Low-Latency Speech Translation

2.3.1. Partial Hypothesis Selection

Latency is not the only criterion in low-latency speech translation. The translation quality is also an important factor. Therefore, low-latency speech translation techniques, such as incremental decoding, must maintain high translation quality while reducing delay.

Partial hypothesis selection from [6] suggests that the model should be selective with the output predictions at each chunk step, as earlier predictions lack sufficient context information about the spoken utterance. Moreover, the acoustic information towards the chunk-endpoints is uncertain, increasing the likelihood of incorrect token predictions at the end of the generated output text. These problems can lead to the generation of unstable predictions at each chunk step. Conditioning the decoder on unstable predictions may trigger a chain of low-quality translationsss at different chunk steps.

To address this problem, the conditioning predictions are carefully selected using policies like hold-n, wait-k and local agreement which are explained in the previous section.

The results from [6] show that local agreement outperforms the aforementioned policies in terms of accuracy-latency trade-off of online transcription. Compared to the offline transcription results, it reduces latency **by 3.8 seconds** at the cost of **0.9% additional WER**.

2.3.2. Multi-Task Learning with Partial and Complete Sentence Translation

The training data for speech translation models typically consists of complete sentence translations. As a result, the model is not exposed to the translation of partial sentences which is essential in low-latency speech translation. This lack of exposure in this area leads to a performance drop in low-latency speech translation.

By using multi-task learning as explained in [13], the model can be trained on a balanced distribution of partial and complete sentences to enhance the accuracy of partial sentence translation while preserving the accuracy of complete sentence translations.

2.3.3. Unobserved Future-Context Prediction

2.3.3.1. Future-Context in Simultaneous Machine Translation

The paper [16] proposes **Translation by Anticipating Future (TAF)**, a method for improving the quality and retaining the latency of simultaneous machine translation systems by incorporating predicted future-context into the machine translation. In essence, a separate large language model (LLM) is used to predict multiple possible continuations of a partial source sentence. In the next step, each predicted continuation is appended to the source prefix and the MT model generates translations from each one of them. Then, the TAF applies a majority voting mechanism to select the most agreed-upon prediction. Finally, the selected prediction is committed, if the following condition with a pre-defined

threshold $\tau \in [0, 1]$ is met:

$$\frac{\text{Count(MAP)}}{n} \geq \tau$$

, where MAP stands for the most agreed-upon prediction and Count(MAP) is the number of occurrences of the MAP within the translation candidates. n is the number of candidates generated by the LLM. If the condition is not met, nothing is committed and the next source segment is read. This method is inspired by the RALCP [24] policy and ensures that a committed prediction is the common prefix of at least a certain number of translation candidates.

When combined with the existing policies such as SM2 [26], Wait-k-Stride-N [27], and Local Agreement [6], the TAF policy improves the translation quality by up to 5 BLEU in comparison to existing policies at the same latency across four translation directions, including German-English, English-Chinese, Chinese-English, and English-Japanese [16]. It is also shown that the TAF improves the translation quality by up to 5 BLEU in comparison to the RALCP policy at the same latency on all combinations of the LLM and MT models [16]. Furthermore, increasing the sample length, the number of future tokens predicted by LLM for each candidate, and the number of candidates to a certain degree improves the quality-latency trade-off of the TAF [16].

These results show that the TAF is an effective policy to improve the quality-latency trade-off of simultaneous machine translation systems. However, this work focuses only on text-to-text translation. Besides, the improvement is observed only on the translation quality, rather than the latency. Therefore, further work is necessary to investigate the effect of the future-context prediction on low-latency speech translation performance, particularly in terms of system latency.

2.4. Large Language Models

Large language models (LLMs) are deep learning algorithms that can recognize, summarize, translate, predict, and generate content using very large datasets [12]. They are pretrained on vast amounts of textual data to perform natural language processing (NLP) tasks and can be finetuned on additional data to specialize in specific domains and tasks [12]. During training, their weights are adjusted so that they learn to make better predictions. The adjustment is done using techniques like backpropagation and weight gradient.

The input of LLMs can be natural language texts, code parts, and structured data such as JSON. Pretrained tokenizers are used to split text into meaningful small units which are called tokens and have unique values called token ids [12]. Each token id is mapped to a learned embedding vector which is the numerical representation of a token. Instead of processing the tokens directly, LLMs process these token embeddings. The output of LLMs is a probabilistic distribution over its vocabulary, which is a collection of tokens. Based on the decoding strategy, a specific number of tokens with the highest probabilities are selected as output.

LLMs have several features which we can utilize for enhanced speech translation performance. One of the most fundamental features of LLMs is that they are trained to predict the next tokens in a given input sequence. In the context of low-latency speech translation, this

feature can be used to predict the next words within the partial transcriptions/translations. Additionally, the output of LLMs being a probabilistic distribution over all tokens in its vocabulary makes it easier to determine whether the predicted future-context is likely to be true or false. Predictions with lower probabilities can be considered low-confidence predictions and be filtered out, which prevents the output from being destabilized.

State-of-the-art LLMs are built on the Transformer [22] architecture which is based on an encoder-decoder architecture. The Transformer architecture does not use recurrent or convolutional layers to know about the order of the tokens within the sequence. Instead, positional encoding of tokens are generated and added to input embeddings. Both have same dimensions so that the two can be summed [22].

One of the key strengths of this architecture is its self-attention mechanism, which allows for weighing the importance of each token in a sequence relative to every other token, thereby improving the model's understanding of the overall context. In this approach, each token embedding, the sum of semantic and positional encoding of a token, is mapped into three different vectors: **query**, **key** and **value**. Using these vectors, the mechanism calculates weights representing the attention of tokens to each other. At the last step, a new representation of the input sequence is calculated as a weighted sum of the values from the value vectors [22], [12]. Understanding the overall context is crucial for both machine translation and future-context prediction, as the predicted output depends on both the current input and the previously generated context. Therefore, the system must effectively capture this information.

Below is the most common NLP tasks performed by LLMs:

- **Machine Translation:** the study of how to use computers to translate texts from one natural language to another [23].
- **Named Entity Recognition:** a NLP task determining which items in the text relates to proper names [4].
- **Question Answering:** a NLP task, which aims to provide precise answers in response to the users' questions in natural language [28].
- **Sentiment Analysis:** the process of analyzing large volumes of text to determine whether it expresses a positive sentiment, a negative sentiment or a neutral sentiment [4].
- **Text Generation**: the process of automatically producing coherent and meaningful text, which can be in the form of sentences, paragraphs or even entire documents [4].
- **Text Summarization:** a NLP task for producing an understandable summary of a set of text [4].

Machine translation (MT) is the most related task to speech translation (ST) among the listed NLP tasks. The key difference is the form of the input: MT translates text, whereas ST translates speech. However, this work focuses on a cascaded implementation of low-latency speech translation, where an ASR and a MT model are used together. In

this regard, LLMs can be used as MT models inside of this cascaded system. Apart from that, LLMs can also be used for future-context prediction of partial translations, as they are trained for predicting the next tokens in a given input text.

3. Methodology

In this section, research methods and techniques are presented.

3.1. Cascaded System Architecture

The speech translation is implemented based on the cascaded system architecture where the ASR and MT models are separate from each other but sequentially integrated. One of its main benefits is the enhanced modularity. Each model can be trained, tested and modified independently from each other [2]. This enables us to build a speech translation which is easier to manage. Moreover, cascaded systems often outperform the E2E systems in terms of speech translation quality, even tough the performance of E2E models are improving thanks to techniques like knowledge distillation [8]. This is due to the fact that a cascaded system allows for independent optimization of the ASR and MT tasks [2]. Lastly, it is easier to localize errors, as the ASR and MT tasks are performed separately.

3.2. Low-Latency Inference

To implement low-latency speech translation, the method of **incremental decoding** is used which is explained in subsection 2.2.2. Furthermore, **partial hypothesis selection** is used to select outputs from each segment step, thereby making the predictions more stable. We adopt the **wait-k strategy** from [6], which is one of the concrete implementations of partial hypothesis selection. The strategy follows a simple algorithm: the system begins processing the partial input sequence as soon as the first k input tokens are received. Partial outputs from each step are then committed at a fixed rate which is defined as the output rate. The following formula describes how the algorithm works:

$$\text{PREFIX}(W^{(c)}) = \begin{cases} \emptyset & \text{if } c \leq k \\ W^{(c)}_{\min(|W^{(c)}|,r)} & \text{if } c > k \end{cases}$$

where **PREFIX()** gives the selected part of the output to commit and $W^{(c)}$ is the output tokens from the c-th chunk/segment step.

3.3. Unobserved Future-Context Prediction

3.3.1. Motivation

One of the drawbacks of low-latency speech translation is that the system has poor context information about input sequence in the early stages of incremental decoding. Starting the translation too early results in low translation quality and accuracy. To mitigate the problems caused by this issue, the system needs to wait for several steps to receive enough input context before beginning the translation. However, this approach increases delay and latency which undermines the primary goal of low-latency speech translation [16].

Predicting unobserved future-context with a LLM may provide the system with additional context information about the input sequence, particularly in the early stages. As a result, the system can begin translating the input sequence earlier without significantly degrading the translation quality.

3.3.2. Prediction Before/After MT

In the cascaded system, there are two approaches for predicting the unobserved future-context. In the first approach, the future-context is predicted by a LLM after each ASR segment step. The model receives the ASR output, which is committed by the system so far at the current step, and predicts the next tokens of the given sequence. Selected future-context tokens are committed at each step along with the selected ASR output tokens. Another approach is to predict the future-context after each MT segment step. The LLM receives the MT translation tokens committed by the system so far as input and predicts the next tokens. These predicted tokens are then selected and committed along with the selected MT output tokens at the end of the segment step.

3.3.3. Providing Fixed-Size Previous Context

Previously translated sentences can serve as additional context for improving future-context prediction accuracy, as they are often semantically related to the current input sentence. Therefore, we adopt a fixed-size window approach in which a fixed number of recently translated sentences are provided as additional context to a selected LLM. These sentences are called previous context. The reason for using a fixed-size approach is to avoid overloading the system with too much information and maintain low inference time.

3.3.4. Prediction Probability Threshold

Incorporating prediction probability thresholds plays a crucial role in the process of selecting unobserved future-context tokens in low-latency speech translation systems. A key challenge in such systems is that not all predictions made by the model are equally reliable. Some tokens predicted by the model may have a low probability, leading to less confidence in their correctness. Inaccurate predictions can negatively impact the overall translation quality, especially in real-time applications.

To address this, a probability threshold is introduced to filter out low-confidence predictions. By setting a minimum threshold for the prediction probability, the system can ensure that only those future-context predictions which the model is confident about are selected and committed. This approach helps to prevent the introduction of uncertain predictions that could degrade the translation quality.

The selection of an optimal threshold value is essential, as too low of a threshold may allow incorrect predictions, while too high of a threshold may result in fewer predictions being selected, reducing the amount of context available for the translation task. The threshold is typically adjusted based on empirical evaluations, with the goal of achieving a balance between maintaining high translation quality and minimizing the latency impact.

In summary, the prediction probability threshold serves as a control mechanism to ensure that only reliable predictions are included in the system's future-context predictions, thereby improving both the stability and accuracy of the low-latency speech translation system.

4. Experimental Setup

4.1. Models

In order to implement low-latency speech translation using the methods explained in the previous chapter, different models are used in the following areas:

4.1.1. Automatic Speech Recognition (ASR)

For low-latency speech translation, we need to use a high-performance ASR model, which requires a state-of-the-art architecture such as the Transformer. Such models are capable of capturing long-range dependencies in audio and producing accurate transcriptions quickly. To ensure reliable performance across various real-world conditions, the ASR model must be robust to background noise and diverse speaker accents. Additionally, it should be trained on a large and diverse dataset to generalize well across different domains and speaking styles.

For these reasons, Whisper[1] [21] is used. It is a Transformer-based model which is trained on multiple tasks, including multilingual speech recognition, speech translation, spoken language identification, and voice activity detection.

Whisper comes with six models, each varying in model size and inference speed. In this work, the **whisper turbo**[2] model is used, as it is the optimized version of **whisper large-v3**[3], the largest model in the Whisper family. It is eight times faster than the **whisper large-v3** model.

Moreover, Whisper uses a sequence of tokens as the training format to handle all speech processing tasks, such as language identification, voice activity detection, with a single model. These tasks are represented with special tokens. In addition, there is a language token which represents the language used in the audio. The sequence also contains text-tokens and timestamp tokens. (See Figure 4.1 for detailed information of Whisper's architecture)

Last but not least, the performance of Whisper models varies depending on the language. Since this work focuses on the evaluation of English-to-German speech translation, it is important to use an ASR model which performs well in English. On the "CommonVoice 15" dataset, the **whisper large-v3** model achieves a word error rate (WER) of 9.3% across various tasks. On the "FLEURS" dataset, the WER is measured at 4.1%. Considering that the **whisper turbo** model's accuracy is close to that of the **whisper large-v3** model, it can be used for the ASR of English.

[1] https://github.com/openai/whisper
[2] https://huggingface.co/openai/whisper-large-v3-turbo
[3] https://huggingface.co/openai/whisper-large-v3

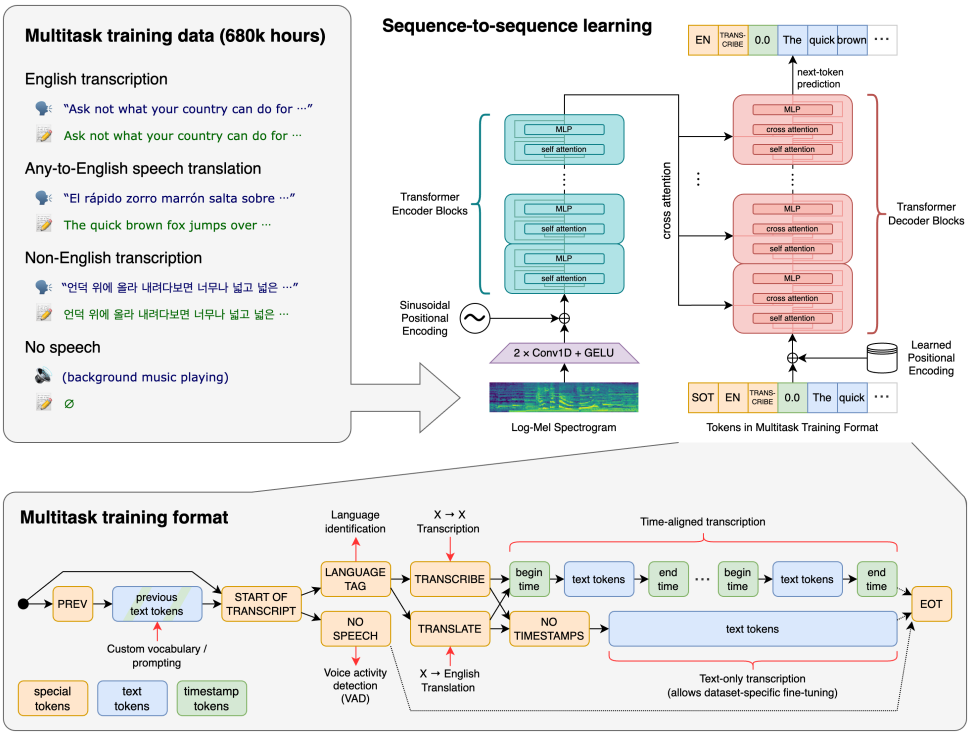


Figure 4.1.: Whisper Architecture
image taken from [21]

4.1.2. Machine Translation (MT)

Low-latency speech translation systems generally produce lower translation quality compared to offline systems. To minimize this quality loss, a high-performance MT model must be used. In this regard, neural machine translation models, which are based on deep learning algorithms, are the state-of-the-art solution for machine translation [13]. Moreover, our low-latency speech translation system utilizes incremental decoding. In this approach, the current translation is conditioned on the received input segments and previously committed translation. Therefore, the system's MT model must be autoregressive. Besides, the model should not attempt to retranslate parts of the input that have already been translated. Therefore, it must be aware of the point from which to continue the translation in the current incremental decoding step. Last but not least, we need to use a distilled lightweight MT model which can perform high-quality translation at low inference time.

For these reasons, **NLLB-200-distilled-600M**[4] is used as the system's MT model which is a distilled version of NLLB-200[5] (No-Language-Left-Behind) [14]. NLLB-200 is an open-source multilingual neural machine translation model which is based on Transformer architecture and developed by Meta AI. It supports the translation of 200 languages,

[4]https://huggingface.co/facebook/nllb-200-distilled-600M
[5]https://ai.meta.com/blog/nllb-200-high-quality-machine-translation/

including low-resource languages. Furthermore, it translates between languages without relying on any intermediate language.

The reason for using the distilled version, **NLLB-200-distilled-600M**, is that this version can perform inference significantly faster than the original NLLB-200 under limited computational resources. While the original NLLB-200 contains over 54 billion parameters, the distilled version contains approximately 600 million parameters. In other words, the lightweight model is about 90 times smaller in terms of parameter count.

4.1.3. Next-Word Prediction

In order to predict next-words of a given input text as unobserved future-context, two models from the Qwen2.5[6] model family [25] are selected:

- **Qwen2.5-0.5B**[7]
- **Qwen2.5-0.5B-Instruct**[8]

These models are the smallest models in the Qwen2.5 model family in terms of the parameter size. They are chosen in order to minimize the latency, which is crucial in low-latency speech translation. In the experiment, both models are evaluated on the next-word prediction. The results of the evaluation are then compared to determine which model should be used in the low-latency speech translation system for predicting the future-context.

The Qwen2.5 model family is a decoder-only LLM family which is based on the Transformer architecture. It supports multilingual capabilities across more than 29 languages, including English and German. Moreover, all models are trained on a dataset which contains over 18 trillion tokens.

4.2. Decoding Configurations

4.2.1. Automatic Speech Recognition

The values for both **Prompt** and **prefix** parameters are set to the transcription generated so far so that the decoder receives the output generated so far and continues the transcription from that point. **The language** is set to the source language manually. Additionally, **half-precision** is enabled for faster decoding. The rest of the configurations are set to the default values defined for the model.

4.2.2. Machine Translation

forced_bos_token_id is set to the token id of the target language, forcing the decoder to generate output in the target language. Moreover, **max_new_tokens**, the maximum

[6]https://github.com/QwenLM/Qwen3
[7]https://huggingface.co/Qwen/Qwen2.5-0.5B
[8]https://huggingface.co/Qwen/Qwen2.5-0.5B-Instruct

number of output tokens to predict,is set to **50** to limit the model's inference at each step while ensuring sufficient token generation in line with the highest output rate value used by the MT model which is 3. The rest of the configurations are set to the default values defined for the model.

4.2.3. Next-Word Prediction

max_new_tokens value is determined differently depending on the future-context being predicted in token-level or word-level. If the prediction is in token-level, the parameter is set to the future-context prediction size. Otherwise, the future-context prediction size is multiplied by a predefined **token_size_per_word** (number of tokens to be predicted for word-level prediction, it is set to 4 by default, as most of the words are tokenized into less than 5 tokens) value and the result is assigned to the parameter. Apart from that, the default system prompt is used for both models. As the user prompt, the base model is only provided with the partial text to complete, whereas the instruction-tuned model receives an additional instruction prefix: **"Predict the next token for the incomplete sentence:"**. The rest of the configurations is set to default values defined for the model.

4.3. Data

To evaluate the performance of low-latency speech translation, the Europarl-ST v1.1[9] [3] is used. It is a multilingual dataset for speech translation. It contains paired audio-text data samples derived from European Parliament debates. It supports nine European languages, including English and German, across 72 translation directions. Since the data is based on political speeches, it covers a broad range of topics. Additionally, the dataset features spoken utterances from a diverse set of speakers, making it a realistic and challenging benchmark.

4.3.1. General Structure

The dataset has four data subsets for each translation direction:

- **train:** training set
- **train-noisy:** training set with noisy audio samples
- **dev:** validation set
- **test:** test set

It includes only unsegmented audio samples. The information required to extract individual audio segments is stored in a separate file, which specifies the original audio filename, as well as the start and end times of each segment. Therefore, the audio segments must be generated by following the segment informations. In addition to that, the dataset provides corresponding reference transcriptions and reference translations for each audio segment.

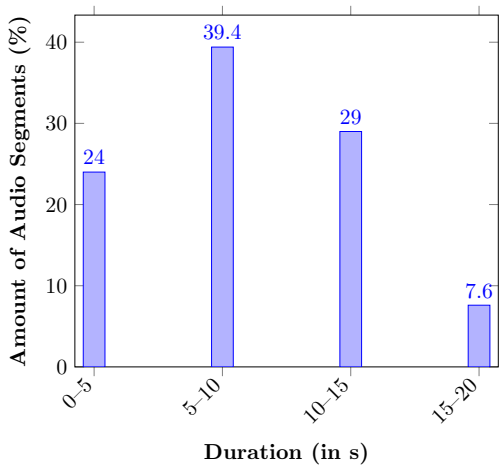
[9]https://www.mllp.upv.es/europarl-st/

4.3.2. Datasets for Evaluation

This work focuses on the English-to-German speech-to-text translation. That's why only samples from this translation direction is used for the evaluation. The evaluation dataset consists of two separate sets: validation set for hyperparameter tuning and test set for unbiased performance evaluation. Both are subsets of Europarl-ST's test set for the aforementioned translation direction.

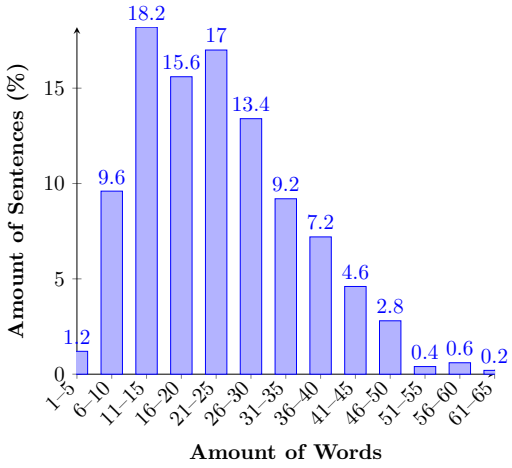
4.3.2.1. Validation Set

It consists of the first 500 audio segments of the **test set** of English-German speech-to-text translation and their corresponding reference transcriptions and translations. Its total length is around 80 minutes. Figure 4.2 shows the length distribution of the validation set. Seeing the length distribution is important, as the speech translation system is evaluated on partial transcription, translation and next-word prediction for varying input sizes.



(a) Length distribution of the audio segments from the validation set by duration in seconds

The upper bound of each interval is exclusive.

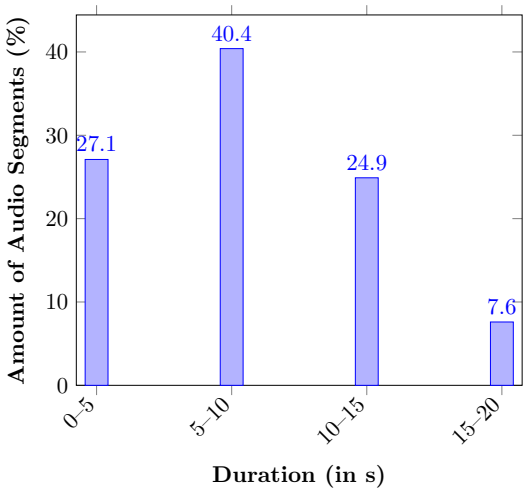


(b) Length distribution of the reference transcription sentences from the validation set by amount of words.

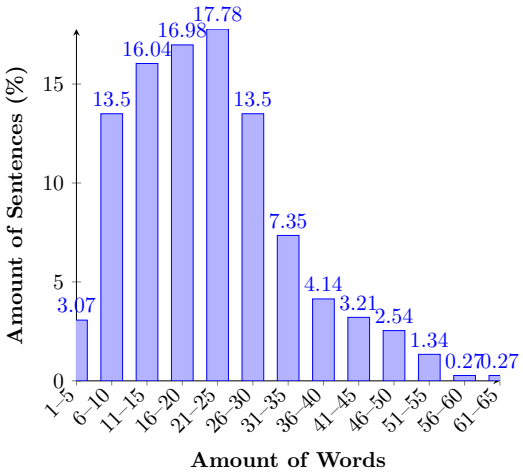
Figure 4.2.: Length distribution of the validation set.

4.3.2.2. Test Set

It consists of the last 753 audio segments of the **test set** of English-German speech-to-text translation and their corresponding reference transcriptions and translations. Its total length is around 100 minutes. Figure 4.3 shows the length distribution of the test set. Seeing the length distribution is important, as the speech translation system is evaluated on partial transcription, translation and next-word prediction for varying input sizes.



(a) Length distribution of the audio segments from the test set by duration in seconds.
The upper bound of each interval is exclusive.



(b) Length distribution of the reference transcription sentences from the test set by amount of words.

Figure 4.3.: Length distribution of the test set.

4.4. Evaluation Metrics

To evaluate the performance of a cascaded low-latency speech translation system, two key aspects must be measured: output quality and latency of the system.

Since this work also involves future-context prediction, its performance must be measured as well.

4.4.1. Word Error Rate (WER)

Word error rate is a common metric used to measure the accuracy of ASR and machine translation outputs by comparing them to reference texts. In this work, WER is used to measure ASR performance. The metric ranges from 0 to 1, where 0 indicates a perfect match and 1 indicates that the compared sentences are completely different. It is calculated as shown below:

$$\text{WER} = \frac{S + D + I}{N}$$

where S, D, and I are the number of substitutions, deletions, and insertions respectively, and N is the total number of words in the reference transcript.

To measure the WER, **jiwer**[10] is used, a Python package developed to evaluate ASR systems. It supports the calculation of various metrics, such as word error rate (WER) and word information preserved. In this work, we focus on the WER, as it is the most commonly used metric to evaluate ASR performance.

4.4.2. Bilingual Evaluation Understudy (BLEU)

BLEU [18] is a commonly used string-based metric to measure quality of machine translation. It does so by measuring the overlap between the MT output and a set of reference translations using n-gram precision. For each n-gram (1-gram, 2-gram, etc.), the precision is calculated as the ratio of the number of matching n-grams in the MT output and the reference translations to the total number of n-grams in the MT output. BLEU score ranges from 0 to 1 (or 0 to 100), with a higher score indicating better translation quality. For a detailed explanation of the BLEU score, please see [18].

To measure the BLEU score, **SacreBLEU**[11] [20] is used, a Python package for evaluating MT systems. It computes BLEU scores which are shareable and comparable. It also supports using multiple reference translations.

4.4.3. Success Rate

We define success rate (SR) to evaluate the performance of future-context prediction. It is calculated as shown below:

$$\text{SR} = \frac{M}{T}$$

[10] https://github.com/jitsi/jiwer
[11] https://github.com/mjpost/sacrebleu

where M is the number of matches and T is the total number of predictions. A match is counted only if all tokens (words) generated by a prediction exactly match with the corresponding ground-truth tokens (words). A word match is counted only if all tokens of that word are predicted correctly.

4.4.4. Average Lagging

In this work, a version of Average Lagging (AL), which is introduced in [11], is used to measure the latency of low-latency speech translation systems. It measures the average lagging of the system behind the ideal policy which is defined later in this section. It is length-invariant. Furthermore, AL measures only computation-unaware latency, meaning that it does not take into account the model's computation time for inference.

Before formalizing AL, it is necessary to define some key terms.

4.4.4.1. Latency Formalizations

- $X = [x_1, \ldots, x_{|X|}]$: the source sequence of the speech translation,
- $Y = [y_1, \ldots, y_{|Y|}]$: the output sequence generated by the translation model,
- $Y* = [y*_1, \ldots, y*_{|Y*|}]$: the reference target sequence,
- x_j: a raw audio segment of duration T_j,
- $X_{1:j} = [x_1, \ldots, x_{|j|}]$: the partial source sequence which must be read by the model to predict y_i.

Furthermore, the actual (d_i) and ideal (d_i^*) delay of y_i can be defined as the following:

$$d_i = \sum_{k=1}^{j} T_k$$

$$d_i^* = (i-1) \sum_{j=1}^{|X|} \frac{T_j}{|Y^*|}$$

4.4.4.2. AL Formula

Let $\tau(|X|) = \min_{i|d_i=\sum_{j=1}^{|X|} T_j}$. It is the index of the first target token for which the entire source sequence is read. Then AL for low-latency speech translation can be defined as:

$$AL_{\text{Speech}} = \frac{1}{\tau(|X|)} \sum_{i=1}^{\tau(|X|)} d_i - d_i^*$$

4.5. Hyperparameters

There are a set of hyperparameters that affect the performance of low-latency speech translation with future-context prediction. The system is first tested on the validation set using different hyperparameter values. Then, its performance is evaluated on the test set using the optimal hyperparameter values, which are extracted from the results for the validation set.

4.5.1. Hyperparameter List

- **segment/chunk size:** fixed size of segments which are extracted from the input sequence and provided to the system during incremental decoding. In this work, only one setup for this hyperparameter is selected: 1 second as the segment size for the ASR and 1 word for the MT. These values are chosen for their simplicity.

- **wait-k value:** k-value which is used in the wait-k strategy. Two k-values must be selected: one for ASR, one for MT. Higher k-value usually leads to better output quality, but also higher latency. The k-values used in the evaluations range from 1 to 20 to simulate and compare realistic scenarios for low-latency speech translation.

- **output rate:** the output rate which is used in the wait-k strategy. Just like the k-value, two rates must be selected: one for ASR, one for MT. Higher output rates can reduce latency, but decrease output quality at the same time. In this work, evaluated output rate values range from 1 to 3, as higher values lead to unstable predictions.

- **previous-context size:** the number of sentences that precede the current input sentence. They are called previous context and provided to the future-context prediction model before the current input sentence. Feeding additional context to the model can increase the likelihood of a correct prediction. As the number of previous sentences increases continuously during translation and causes high inference time at some point, only a few context sizes, up to 10 sentences, are selected for the evaluation.

- **future-context prediction size:** the number of tokens (words) to be generated by a prediction. Predicting more tokens (words) may lead to less stable future-context predictions, which can negatively impact the overall performance of the system. The prediction size is limited to a maximum of 3 tokens/words, as predicting even 3 units causes a significant drop in the success rate.

- **prediction probability threshold:** the minimum prediction probability required to accept a future-context prediction. A higher threshold may help filter out incorrect predictions which tend to have lower prediction probabilities. On the other hand, it may increase the latency of the system due to having less selected future-context predictions for the system. To determine the optimal threshold, values ranging from 10 to 100 in increments of 10 are tested.

- **time of future-context prediction:** the time at which future-context prediction is performed. It can be executed either right after each ASR incremental decoding step or at the end of each MT step.

4.5.2. Hyperparameter Setup for Low-Latency Cascaded System

To see the performance of the low-latency cascaded system without future-context prediction on different wait-k setups, the following setups are chosen:

1. wait-2 - output rate=2 **(ASR)** & wait-5 - output rate=2 **(MT)**
2. wait-2 - output rate=2 **(ASR)** & wait-5 - output rate=3 **(MT)**
3. wait-2 - output rate=3 **(ASR)** & wait-5 - output rate=2 **(MT)**
4. wait-2 - output rate=3 **(ASR)** & wait-5 - output rate=3 **(MT)**

These setups are selected based on their superior quality-latency trade-off compared to other setups from subsection 5.2.1 and subsection 5.2.2.

4.6. SimulEval Framework

SimulEval[12] is a framework designed to run and evaluate simultaneous/low-latency translation on text and speech. In addition to this, SimulEval can measure the system's latency and translation quality by using metrics like average lagging, average proportion, WER and BLEU.

[12]https://github.com/facebookresearch/SimulEval

5. Evaluation

In this section, performance evaluation results for different types of low-latency speech translation setups are presented and discussed.

5.1. Speech Translation - Offline Implementation

5.1.1. ASR Results

To establish a benchmark for the system's low-latency performance, the performance of the offline ASR is measured. This refers to the model's performance when it transcribes after receiving the entire input sequence. It serves as the upper bound for the performance of the ASR system on the validation set.

The whisper turbo model achieves a WER of **19.35%** on the validation set. Commonly observed errors include missing the beginning of a sentence, confusion of words with their homophones or near-homophones (i.e., different words with identical or similar pronunciation), and inconsistent use of punctuation.

5.1.2. MT Results

Following the same logic used for evaluating the ASR performance, the MT model's offline performance is measured first.

The **nllb-200-600M** model is used for the MT system as mentioned earlier. To evaluate the offline MT system in isolation, the reference transcriptions provided by the Europarl-ST dataset are used. In this setup, the system achieves a BLEU score of **22.89**.

5.1.3. Cascaded System Results

The last step for evaluating the offline implementation is to measure the performance of the offline cascaded system. To do so, the transcriptions predicted by the offline ASR system are provided to the offline MT model. In this setup, the cascaded system achieves a BLEU score of **19.87**, approximately **3 points lower** than the previous setup from subsection 5.1.2. The lower score compared to the isolated MT performance can be explained by the fact that the transcription provided by the ASR system does not match entirely with the reference transcriptions, and the reference translations are based on the reference transcriptions. This result can be used as a benchmark to evaluate the performance of the cascaded low-latency speech translation system.

5.2. Speech Translation - Low-Latency Implementation Without Future-Context Prediction

5.2.1. ASR Results

In the next step, the performance of the low-latency ASR system is evaluated. This setup does not include future-context prediction. Later, the results for this setup can be used for comparison to evaluate the effect of future-context prediction on the performance of the low-latency ASR system.

To run and evaluate this setup, the SimulEval framework is used, and an agent is implemented which uses the whisper **turbo** model and wait-k strategy.

The model receives audio segments from the validation set. As the results from Figure 5.1 show, the average lagging (AL) decreases with lower k-values when the output rate is fixed. However, lowering the k-value increases the word error rate (WER), resulting in lower output quality. For example, the wait-2 strategy with an output rate of 2 achieves a WER of approx. **24%** and AL of approx. **2.5 seconds**. It means an increase of the WER by **24%** and a decrease of the AL by **70%** compared to the offline ASR performance (using the AL of the wait-1000 setup as the AL for the offline ASR latency baseline). Moreover, it can be concluded that output rate has also a significant impact on the WER and AL when the same k-value is used. Increasing the output rate reduces latency while decreasing the output quality. Another finding is that the system's performance sensitivity to the output rate increases when smaller k-values are used.

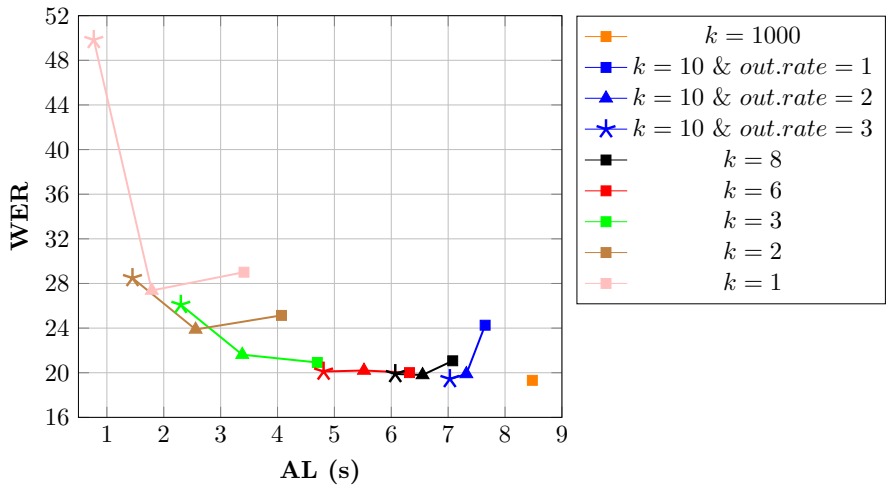


Figure 5.1.: WER (word error rate) and AL (average lagging) results for setups with different wait-k strategies used for the low-latency ASR with whisper turbo
In the legend, **k** represents wait-k value for the ASR system and out.rate stands for output rate for the wait-k strategy. Setups with the same marker shape have identical output rates. For example, setups with square has an output rate of 1.

5.2.2. MT Results

It is also necessary to evaluate the MT system in the same way as the ASR system, as explained in the previous section. This time, the segment size is chosen as 1 token and the model receives the reference transcriptions provided by the Europarl-ST dataset. The **nllb-200-600M** model is used as the MT model.

Looking at Figure 5.2, it can be observed that using smaller k-values while fixing the output rate leads to decreased latency and reduced translation quality at the same time. Similarly, increasing the output rate while keeping the k-value fixed results in the same effect. In essence, one needs to compromise translation quality in order to reduce latency.

To see the effect of low-latency MT on the performance, let's compare the results for the wait-3 and wait-1000 setup (using an output rate of 2). Again, the wait-1000 setup serves as baseline for the performance of the offline MT system. With the former setup, the model achieves an AL of approximately **2**, down from **23 tokens**. However, the wait-2 setup achieves a BLEU score of approximately **13**, which is almost **10** points lower than the score of the wait-1000 setup (**91.3%** decrease in latency and **43%** lower quality).

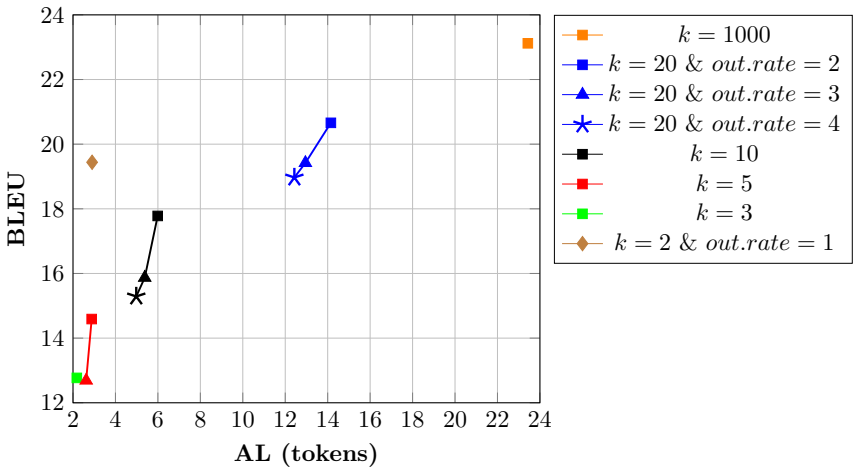


Figure 5.2.: BLEU and AL (average lagging) results for setups with different wait-k strategies used for the low-latency MT with the **nllb-200-600M** model

5.2.3. Cascaded System Results

Finally, the cascaded model is evaluated by using the same ASR and MT models from the previous sections with the same segment sizes. The chosen setups for the evaluation are listed in subsection 4.5.2.

The results are illustrated in Figure 5.3. Increasing only the output rate for the MT decreases both latency and translation quality. Similarly, increasing only the output rate for the ASR results in the same effect, although the change in the translation quality is more moderate.

Setups 1 and 3 perform better than the second setup from subsection 5.1.2 (which achieves a BLEU score of **19.87**) in terms of translation quality, whereas setups 2 and 4

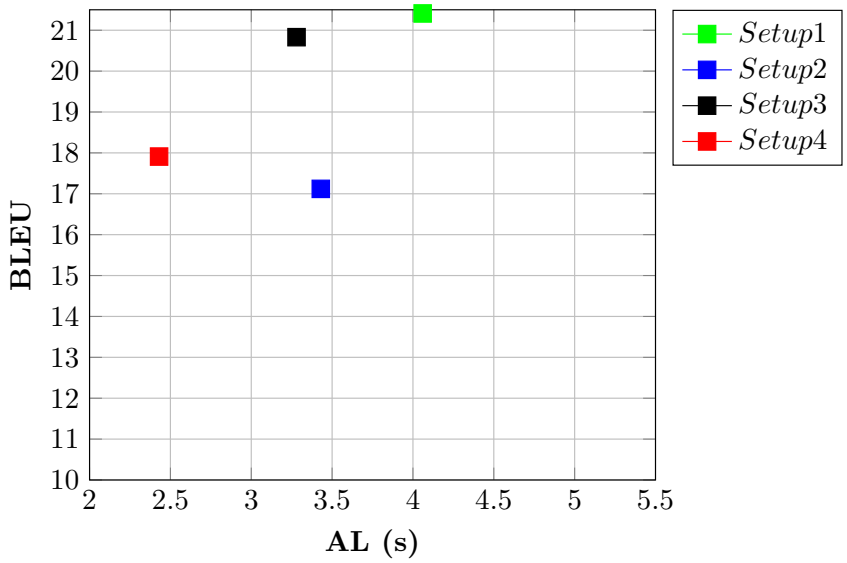


Figure 5.3.: AL (average lagging) and BLEU score results for different low-latency cascaded model setups

perform worse in this regard. However, even the worst-performing setup achieves a BLEU score of approximately **17** which is only **2.87 points lower** than the translation quality of the offline MT setup.

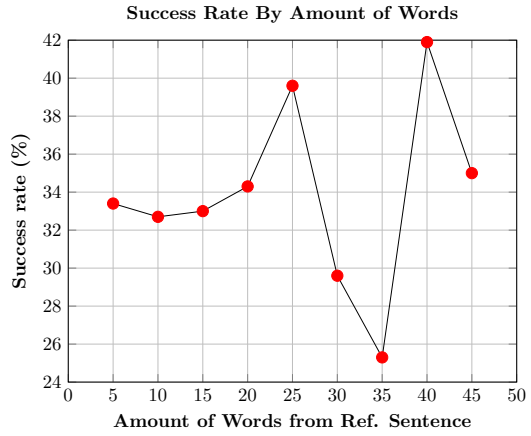
5.3. Future-Context Prediction on ASR Reference Sentences

Before integrating future-context prediction into the low-latency cascaded speech translation model, the optimal hyperparameter values must be found. Furthermore, it must be decided which model from subsection 4.1.3 should be used in the cascaded model for future-context prediction. To evaluate the performance of the models, reference transcriptions from the validation set are used.

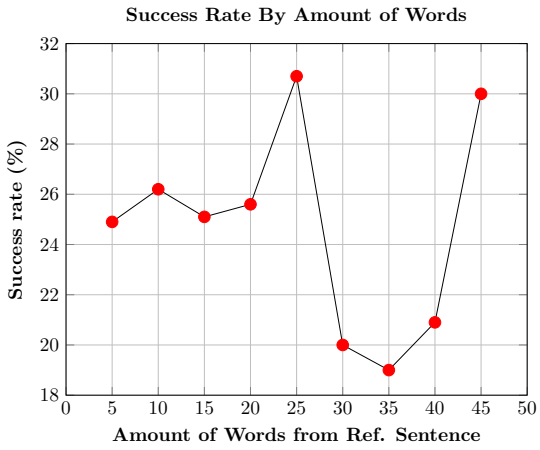
5.3.1. Impact of Input Size On Performance

It is important to decide on the time in which the future-context prediction is used in the low-latency speech translation, as predicting in the earlier stages may result in unstable predictions, which can affect the translation quality negatively.

In Figure 5.4, the success rate of the future-context prediction based on the input size is illustrated for both base model and instruction-tuned model. The input size is the number of words from the input sentence provided to the model. For this evaluation, only the next word is predicted and five previous sentences are provided before the current input sentence as previous context. Sentences which don't have more words than the examined input size are excluded from that part of the evaluation, as there is not next word to be predicted for them.



(a) Results for the Qwen2.5-0.5B base model



(b) Results for the Qwen2.5-0.5B-Instruct instruction-tuned model

Figure 5.4.: Success rate of future-context prediction by number of words provided as input from the reference sentences of the validation set
Only the next word is predicted. Maximum " of sentences us as previous context is 5.

The results show that the success rates achieved by the base model are approximately 8% higher than the ones achieved by the instruction-tuned model. It means that the base model performs better in the future-context prediction task. Therefore, the base model can be used in the cascaded model to optimize the performance of the low-latency speech translation with future-context prediction.

Another finding is that no correlation is observed between the size of the current input sentence and the success rate. For example, the success rate increases with larger input size in the interval 10-25. However, the success rate for the input size 5 is higher than the success rate for the input size 30 in both models. Therefore, the future-context prediction can be used in each stage of the low-latency speech translation, without waiting for the model to receive a specific number of input segments. By starting to use the future-context prediction from the very beginning, system latency can be reduced further.

5.3.2. Impact of Previous-Context Size On Performance

As mentioned earlier in subsection 3.3.3, using previous sentences before the sentence to be predicted can provide the model additional context, thus increasing its success on predicting the future-context. However, using large number of previous sentences increases the number of input tokens and may slow down the model's inference. Additionally, it increases the computational cost, as more tokens need to be processed by the model. Therefore, adopting a fixed-size previous context approach can be more efficient in terms of the system's latency.

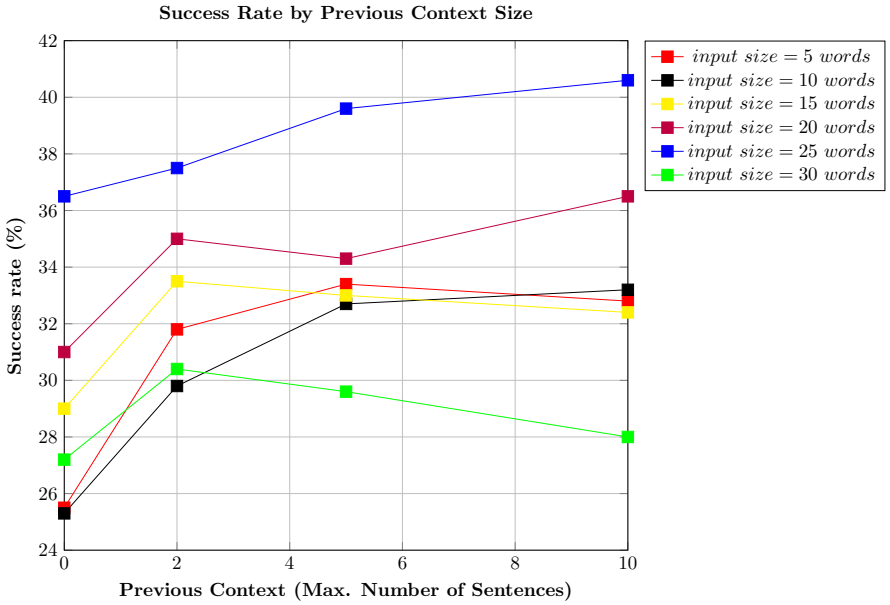


Figure 5.5.: Success rate of future-context prediction of the Qwen2.5-0.5B base model by maximum number of sentences provided as previous context
Only the next word is predicted.

Figure 5.5 shows the success rate of the base model based on the size of the previous context. The evaluation is performed using different number of words from the current input sentence. In general, the success rate increases significantly up to a previous context size of 5, especially when predicting the earlier words in the sentence. Doubling the previous context to 10 leads to only a slight increase in the success rate, and in some cases, the prediction performance even decreases. One factor leading to this result can be that the contents of the previous sentences and the current sentence begins to differ as the range of previous context increases. Therefore, utilizing more previous sentences may not help with predicting the current context. As a result, previous context is set to the last 5 sentences preceding the current input sentence.

5.3.3. Impact of Future-Context Prediction Size On Performance

Another hyperparameter that needs to be considered is the prediction size. It means the number of next words which the model must predict in a single prediction. Predicting more words provides the model with more context in the earlier stages, thus reducing the system's latency. However, it becomes harder for the model to predict correctly as the prediction size grows which can effect the quality of the resulting output negatively.

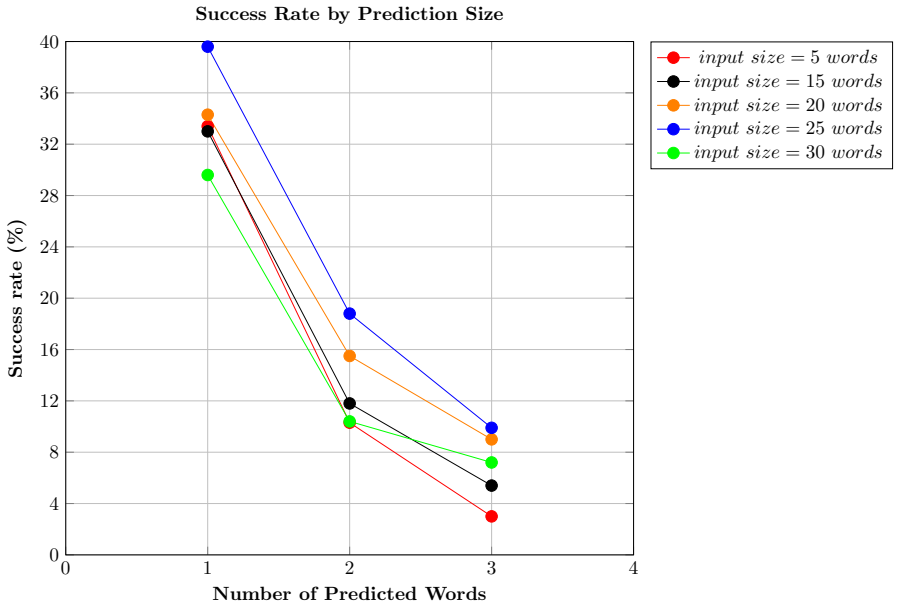
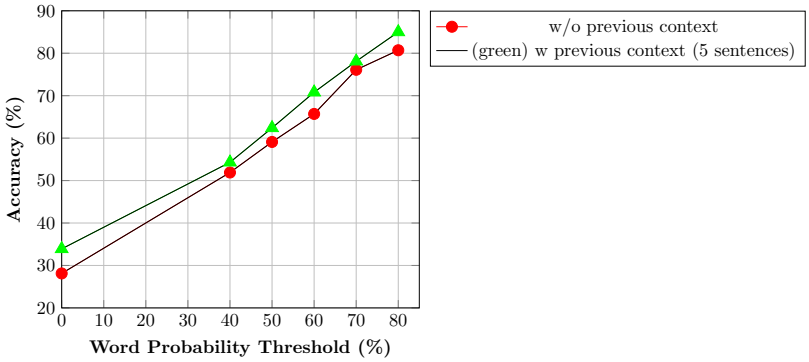


Figure 5.6.: Success rate of future-context prediction of the Qwen2.5-0.5B base model by the number of words predicted as future-context
Maximum number of previous sentences used as previous context is 5.

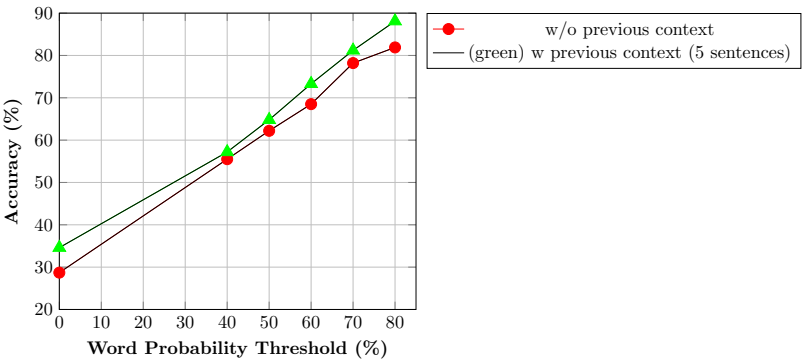
In Figure 5.6, the success rate of the base model is shown based on the number of predicted words. Similar to the evaluation in subsection 5.3.2, the performance of the model is evaluated on different input sizes. For each input size, the success rate drops drastically when more than one word is predicted. For example, the success rate of the base model drops **from approximately 40% to below 20%** when predicting the next two words instead of one only. Predicting the next three words decreases the success rate even further, to approximately **10%**. Therefore, in the following setups, only the very next word is predicted when using future-context prediction in the low-latency speech translation system.

5.3.4. Impact of Prediction Probability Threshold On Performance

Setting the threshold, which is used to filter out the low-confidence future-context predictions, to different values, the sensitivity of the success rate to the threshold can be observed. To do so, reference transcriptions from the validation set and test set are used. Starting from the second word of each sentence, the model predicts the very next word one by one when provided with the preceding words of the sentences. In order to calculate the success rate for a specific threshold value, only the predictions with a probability over the threshold value are compared with the ground-truth words. The probability of a predicted word is calculated as the mean of the probabilities of all tokens making that word.



(a) Results on the validation set

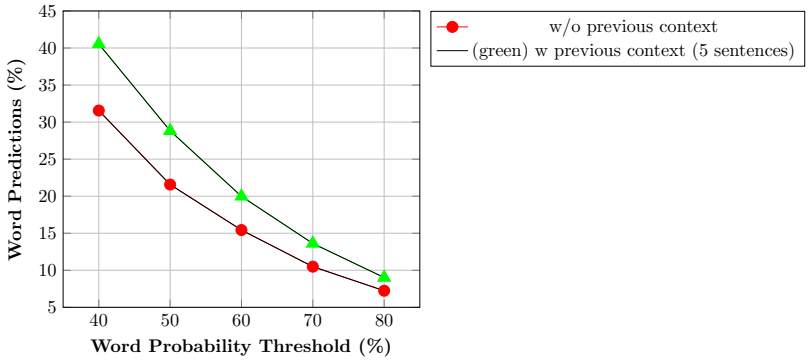


(b) Results on the test set

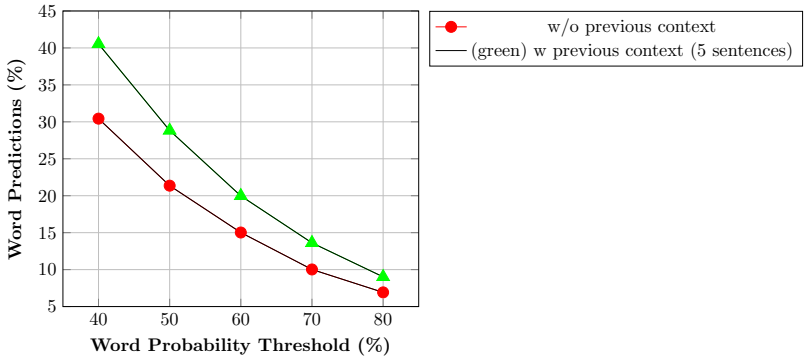
Figure 5.7.: Word prediction accuracy of the Qwen2.5-0.5B base model by different prediction probability threshold values

Predictions with a probability lower than the threshold value are excluded from the accuracy calculation for that threshold. Only the next word is predicted as future-context.

Figure 5.7 illustrates the results of the described evaluation on both the validation and test sets. The results show a positive correlation between the threshold value and word accuracy in both cases. On the test set, the model achieves a word accuracy close to **90% (80%)** with **80% (70%)** threshold when previous context is used. Moreover, using previous context consistently improves accuracy across all threshold values. It can also be observed that the accuracy results on the validation set are close to those on the test set.



(a) Results on the validation set



(b) Results on the test set

Figure 5.8.: Percentage of word predictions which exceeds a specific threshold value
On the validation set, a total of 11 215 words are predicted by the model. It is 15 499 word predictions for the test set.

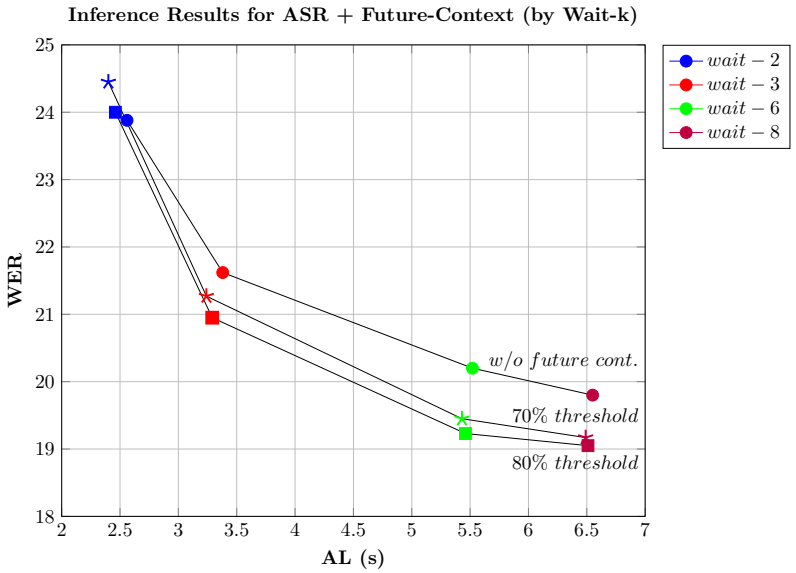
Even though setting the threshold value higher consistently improves accuracy, it is likely that fewer and fewer predictions are validated as the threshold increases. When the model is provided with less predicted future-context, the latency of the cascaded speech translation system may increase. In Figure 5.8, it is shown how much percent of the next word predictions exceed a specific threshold. As expected, the percentage decreases consistently as the threshold value increases. The same trend is observed in the test set as well. On the validation set, approximately **4 out of 10 predictions have a higher probability than 40%**, whereas only about **1 out of 10 predictions exceeds the threshold value of 80%**.

5.4. ASR - Low-Latency Implementation With Future-Context Prediction

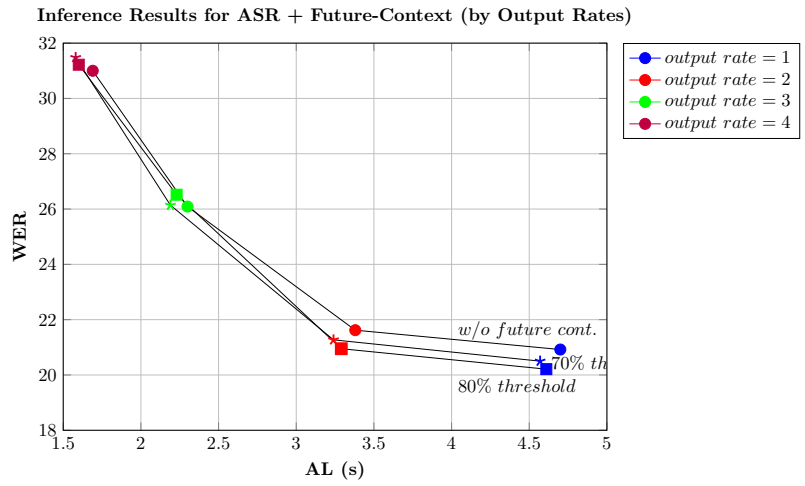
After evaluating the future-context prediction in isolation, the first step is to evaluate the future-context-prediction-backed ASR system. By doing so, the impact of the future-context prediction can be observed on the ASR performance. In this setup, future-context is predicted after each chunk step of the incremental ASR and committed alongside the ASR prediction if it is selected.

Figure 5.9 shows the results of the evaluation for the next word future-context prediction using the validation set. The results of the same evaluation for the token-level future-context prediction can be found in Figure A.2. For this evaluation, three setups are used: ASR without future-context prediction, ASR with future-context prediction using threshold values of 70% and 80% for the prediction probability. In all setups except for the wait-2 setup, w/o future-context setup performs worse than other setups in terms of the output quality. Additionally, using future-context, regardless of which threshold value is used, reduces the latency slightly for each setup. Moreover, **selecting the future-context predictions with the 70% threshold results in higher WER than the one when using 80% as threshold, with a slight improvement in latency**. This can be explained with the fact that the probability of the selected predictions is higher when setting the threshold higher. Such predictions tend to be more stable. However, higher the threshold is, less future-context prediction is selected due to low-confidence predictions being filtered out. With less future-context, the speech translation system does not become faster which increases the latency. Apart from that, the WER gap between the two future-context setups with different thresholds increases as the ASR prediction starts earlier (by using lower k-value for the wait-k).

Figure 5.10 illustrates the results of the same evaluation, but using the test set. Since latency is the priority in low-latency speech translation and the WER results are close for both threshold setups, future-context prediction is performed using only 70% as the threshold value for evaluation of the test set. On the contrary to the results on the validation set, w/o future-context setup performs slightly better for each wait-k setup in terms of the output quality. Furthermore, using future-context reduces the latency slightly for each wait-k setup.

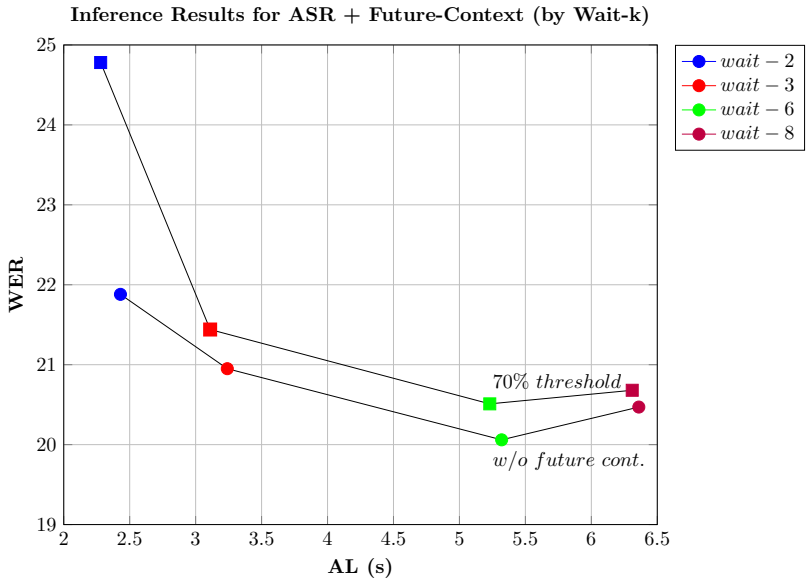


(a) Results by different wait-k values. The output rate is fixed at 2.

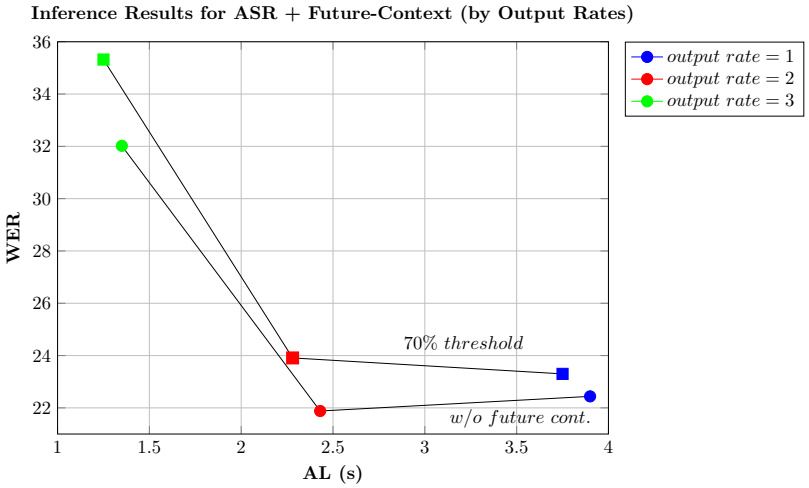


(b) Results by different output rates. The wait-k value is fixed at 2.

Figure 5.9.: WER (word error rate) and AL (average lagging) results for ASR with/without word-level future-context prediction on the validation set
Future-context is predicted by the Qwen2.5-0.5B base model. Previous context size is set to 5 and only the next-word is predicted.



(a) Results by different wait-k values. The output rate is fixed at 2.



(b) Results by different output rates. The wait-k value is fixed at 2.

Figure 5.10.: WER (word error rate) and AL (average lagging) results for ASR with/without word-level future-context prediction on the test set
Future-context is predicted by the Qwen2.5-0.5B base model. Previous context size is set to 5 and only the next-word is predicted.

5.5. Cascaded System - Low-Latency Implementation With Future-Context Prediction

Finally, I evaluate the online cascaded model which performs the entire low-latency speech translation, including the ASR and MT. This evaluation uncovers whether future-context prediction improves the performance of the low-latency speech translation in terms of the latency and translation quality.

Figure 5.11 shows the evaluation results. When the output rate value for the MT is fixed, both future-context setups (using different thresholds of 70% and 80%) improves the latency by a few milliseconds. Also , using a lower threshold value helps with improving the latency even further. These results are observed for all setups with different wait-k values. Apart from that, future-context prediction has a minimal impact on translation quality, with differences remaining below 1 BLEU point. Predicting future-context before MT and using wait-2 strategy even increases the translation quality slightly. Another interesting finding is that selecting future-context with higher probability threshold leads to a slight increase in BLEU score, thus improving the translation quality.

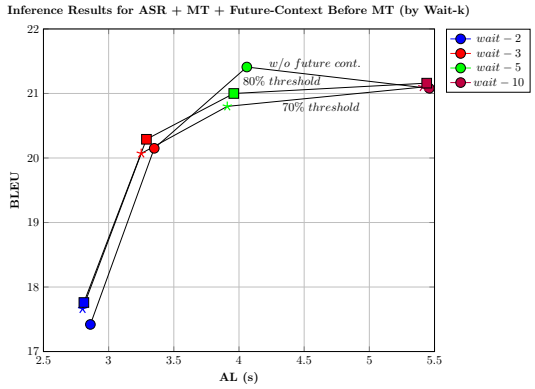
Looking at the result for setups with different output rate values (and fixed value of 2 as the MT wait-k strategy), we observe the same findings which are explained in the last paragraph: using future-context predictions improves the system latency in all setups while the translation quality is preserved.

The graphs in Figure 5.11 also illustrate the impact of the timing of future-context prediction (for next word) on the performance of the low-latency speech translation system, using the validation set. In nearly all setups, predicting the future-context before MT results in higher translation quality than predicting it after MT. This difference is more noticeable when the threshold value for selecting future-context predictions is set to a lower value (here 70%). Several factors may contribute to this finding:

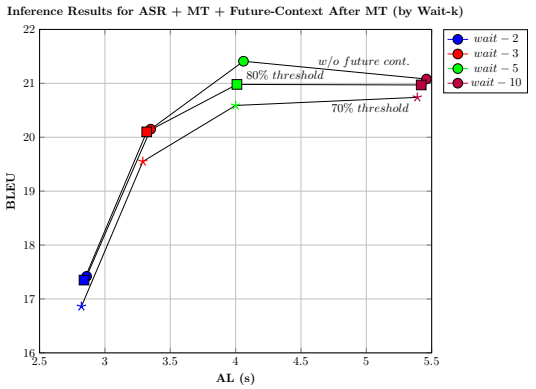
1. When future-context is predicted before MT, the MT model receives the predicted future-context at each step, functioning as an additional input context. Having more context than the "after MT" setup, the MT model may generate better translations in terms of quality and accuracy.

2. Future-context is predicted in English in the "before MT" setup, whereas it is predicted in German in the "after MT" setup. The base model may predict English words better than German words. Therefore, the translation quality may be higher in the "before MT" setup.

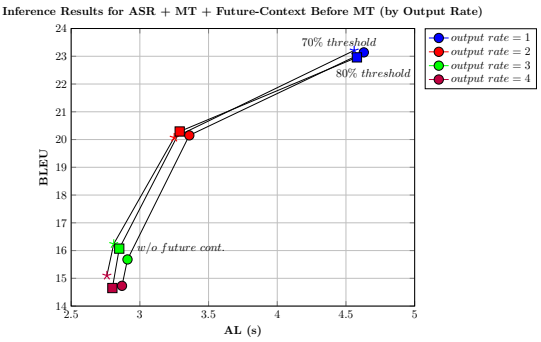
Additionally, the "before MT" future-context prediction improves the latency by a few milliseconds compared to the "after MT" setup.



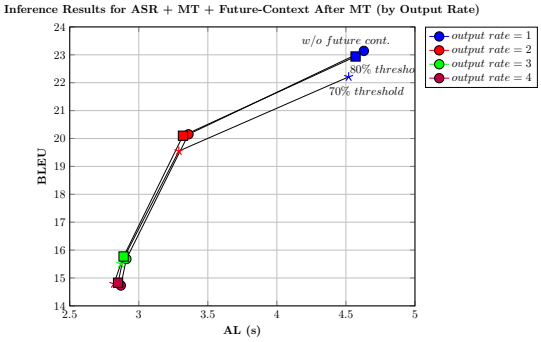
(a) Results for setups with different wait-k values when future-context is predicted right after each ASR chunk step. Output rate for the MT is fixed at 2.



(b) Results for setups with different wait-k values when future-context is predicted right after each MT chunk step. Output rate for the MT is fixed at 2.



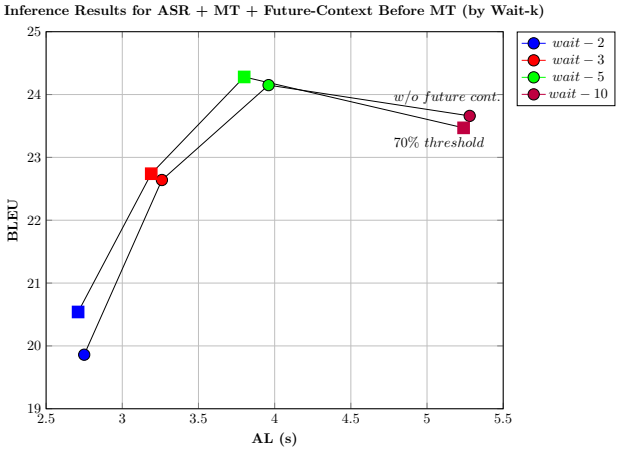
(c) Results for setups with different output rate values when future-context is predicted right after each ASR chunk step. Wait-k value for the MT is fixed at 3.



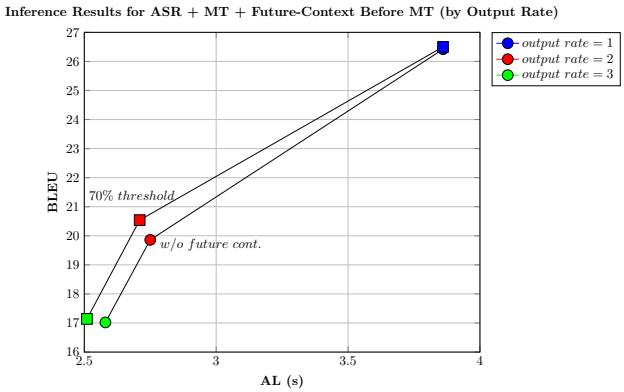
(d) Results for setups with different output rate values when future-context is predicted right after each MT chunk step. Wait-k value for the MT is fixed at 3.

Figure 5.11.: BLEU score vs AL (average lagging) results for low-latency speech translation with/without next word future-context prediction by the Qwen2.5-0.5B base model on the validation set

All setups use the same ASR strategy: wait-2 with output rate 2. Only the next word is predicted. Previous context size is set to 5 sentences.



(a) Results for setups with different wait-k values. Output rate for the MT is fixed at 2.



(b) Results for setups with different output rate values. For al setups, the MT uses wait-2 strategy.

Figure 5.12.: BLEU score vs AL (average lagging) for low-latency speech translation with/without next word future-context prediction by the Qwen2.5-0.5B base model on the test set

ASR strategy is same for all setups: wait-2 with an output rate of 2. Only the next word is predicted. Previous context size is set to 5 sentences.

Lastly, the graphs in Figure 5.12 show the results of the same evaluation, but using the test set. In this evaluation, the performance of the two setups are compared: speech translation without future-context and with future context using a threshold value of 70% to select predictions. In nearly all setups, future-context prediction improves both the latency and translation quality. The difference in BLEU score is lower than 1 point for each setup. The latency improvement is limited with a few milliseconds.

6. Conclusion

In this work, we investigated the potential of using unobserved future-context prediction to improve the performance of low-latency speech translation systems. While traditional approaches to reduce latency often sacrifice translation quality due to insufficient context, this work proposed an alternative: predicting future tokens using lightweight large language models (LLMs) to enrich the contextual input available during early stages of translation.

The implementation was based on a cascaded architecture involving separate ASR and MT components, with Qwen2.5-0.5B serving as the future-context predictor. A series of experiments were conducted to evaluate low-latency speech translation with/without future-context prediction in numerous scenarios. The findings suggest that selecting only high-confidence future-context predictions, especially the very next word, can reduce the system's latency with minimal change in translation quality.

In conclusion, this thesis provides empirical evidence that unobserved future-context prediction is an effective method to address the latency-quality trade-off in low-latency speech translation in the English-German direction.

6.1. Answers to Research Questions

- **Research Question 1: How can unobserved future-context be predicted?**

  Answer: The unobserved future-context can be predicted by lightweight LLMs, particularly by base models such as Qwen2.5-0.5B. To do so, the model receives the so far generated transcription or translation along with recently completed translations as input. The model predicts then the next tokens in the given input sequence.

- **Research Question 2: How can unobserved future-context prediction be integrated into existing low-latency speech translation systems?**

  Answer: The predictions can be generated after each ASR (MT) segment step by providing a fixed-size of previously generated transcriptions (translations) and the so far generated current transcription (translation), including the predicted tokens to commit at the end of the current segment step, as input to the future-context prediction model. In the next step, high-confidence future-context predictions are selected based on a pre-defined probability threshold. Selected future-context tokens are then committed along with the ASR (MT) predictions to commit.

- **Research Question 3: How do different configurations of the future-context prediction module impact low-latency speech translation performance?**

  Answer: One of the most important findings was that the setups predicting future-context before MT (using generated transcriptions as input) outperformed the setups predicting it after MT (using generated translations as input) in almost all scenarios. This shows that the timing of the future-context prediction plays a vital role in low-latency speech translation.

  Another factor impacting the performance of low-latency speech translation systems was found to be the size of previous context. To boost future-context prediction performance, recently completed translations can be used as previous context, adopting a fixed-size window strategy. We noticed that the prediction accuracy increased consistently with increasing size of previous context up to 5 sentences. After including the last 5 sentences however, the impact of additional previous context on the prediction accuracy stagnated.

  Moreover, we found that the probability threshold played also important role in the performance of low-latency speech translation systems. Using higher threshold values, the accuracy of both future-context prediction and low-latency speech translation increased in nearly all scenarios. On the other hand, setups using lower threshold values reduced latency more than the setups with higher threshold values. These results show that its value must be selected carefully to balance the latency-accuracy trade-off of such systems.

  Finally, the most effective prediction was found to be the prediction of the very next word, as predicting more words reduced the accuracy of future-context prediction by more than half.

6.2. Limitations and Future Work

This thesis focused on improving low-latency speech translation using unobserved future-context prediction, but several limitations remain.

First, the experiments were conducted only on the English-German language pair, both of which are high-resource languages. Future work should examine the effectiveness of this approach in low-resource scenarios, where data scarcity may impact prediction accuracy. Second, only the next word was predicted to maintain high accuracy. However, this limits the amount of additional context available. Future research could explore improving multi-word prediction accuracy through better decoding or confidence-based filtering methods. Third, this work uses only the wait-k strategy as the read-write policy. Future studies should also investigate the effect of future-context prediction while using different read-write policies such as local agreement. Finally, although this work used a cascaded system, future studies could explore integrating future-context prediction into end-to-end architectures for potentially greater performance gains, particularly in terms of improved latency.

Bibliography

[1] Qianqian Dong et al. “Learning When to Translate for Streaming Speech”. In: (2022). URL: https://arxiv.org/abs/2109.07368.

[2] Thierry Etchegoyhen et al. “Cascade or Direct Speech Translation? A Case Study”. In: *Applied Sciences* 12.3 (2022). DOI: 10.3390/app12031097. URL: https://www.mdpi.com/2076-3417/12/3/1097.

[3] J. Iranzo-Sánchez et al. “Europarl-ST: A Multilingual Corpus for Speech Translation of Parliamentary Debates”. In: (2020), pp. 8229–8233.

[4] Diksha Khurana et al. “Natural Language Processing: State of The Art, Current Trends and Challenges”. In: (2017). DOI: https://doi.org/10.1007/s11042-022-13428-4. URL: https://arxiv.org/abs/1708.05148.

[5] Dan Liu et al. “Cross Attention Augmented Transducer Networks for Simultaneous Translation”. In: (2021). URL: https://aclanthology.org/2021.emnlp-main.4/.

[6] Danni Liu, Gerasimos Spanakis, and Jan Niehues. “Low-Latency Sequence-to-Sequence Speech Recognition and Translation by Partial Hypothesis Selection”. In: (2020). URL: https://arxiv.org/abs/2005.11185.

[7] Xiaoqian Liu et al. “Recent Advances in End-to-End Simultaneous Speech Translation”. In: (2024). URL: https://arxiv.org/abs/2406.00497v2.

[8] Yuchen Liu et al. “End-to-End Speech Translation with Knowledge Distillation”. In: (2019). URL: https://arxiv.org/pdf/1904.08075.

[9] Mingbo Ma et al. “STACL: Simultaneous Translation with Implicit Anticipation and Controllable Latency using Prefix-to-Prefix Framework”. In: (2019). URL: https://aclanthology.org/P19-1289/.

[10] Xutai Ma, Juan Pino, and Philipp Koehn. “SimulMT to SimulST: Adapting Simultaneous Text Translation to End-to-End Simultaneous Speech Translation”. In: (2020). URL: https://arxiv.org/abs/2011.02048.

[11] Xutai Ma et al. “SimulEval: An Evaluation Toolkit for Simultaneous Translation”. In: (2020). URL: https://arxiv.org/abs/2007.16193.

[12] Humza Naveed et al. “A Comprehensive Overview of Large Language Models”. In: (2024). URL: https://arxiv.org/abs/2307.06435.

[13] Jan Niehues et al. “Low-Latency Neural Speech Translation”. In: (2018). URL: https://arxiv.org/abs/1808.00491.

[14] NLLB Team et al. “No Language Left Behind: Scaling Human-Centered Machine Translation”. In: (2022). URL: https://arxiv.org/abs/2207.04672.

[15] Douglas O'Shaughnessy. "Trends and developments in automatic speech recognition research". In: *Computer Speech Language* 83 (2024), p. 101538. ISSN: 0885-2308. DOI: https://doi.org/10.1016/j.csl.2023.101538. URL: https://www.sciencedirect.com/science/article/pii/S0885230823000578.

[16] Siqi Ouyang et al. "Anticipating Future with Large Language Model for Simultaneous Machine Translation". In: (2024).

[17] Sara Papi, Matteo Negri, and Marco Turchi. "Attention as a Guide for Simultaneous Speech Translation". In: (2023). URL: https://arxiv.org/abs/2212.07850.

[18] Kishore Papineni et al. "BLEU: a Method for Automatic Evaluation of Machine Translation". In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics. 2002, pp. 311–318.

[19] Peter Polák et al. "Incremental Blockwise Beam Search for Simultaneous Speech Translation with Controllable Quality-Latency Tradeoff". In: (2023). URL: https://arxiv.org/abs/2309.11379.

[20] Matt Post. "A Call for Clarity in Reporting BLEU Scores". In: *Proceedings of the Third Conference on Machine Translation: Research Papers*. Belgium, Brussels: Association for Computational Linguistics, Oct. 2018, pp. 186–191. URL: https://www.aclweb.org/anthology/W18-6319.

[21] Alec Radford et al. "Robust Speech Recognition via Large-Scale Weak Supervision". In: (2022). URL: https://arxiv.org/abs/2212.04356.

[22] Ashish Vaswani et al. "Attention Is All You Need". In: *OpenAI Blog* (2023). URL: https://arxiv.org/abs/1706.03762.

[23] Haifeng Wang et al. "Progress in Machine Translation". In: *Engineering* 18 (2022), pp. 143–153. ISSN: 2095-8099. DOI: https://doi.org/10.1016/j.eng.2021.03.023. URL: https://www.sciencedirect.com/science/article/pii/S2095809921002745.

[24] Minghan Wang et al. "Simultaneous Machine Translation with Large Language Models". In: (2024). URL: https://arxiv.org/abs/2309.06706.

[25] An Yang et al. "Qwen2.5 Technical Report". In: *arXiv preprint arXiv:2412.15115* (2024).

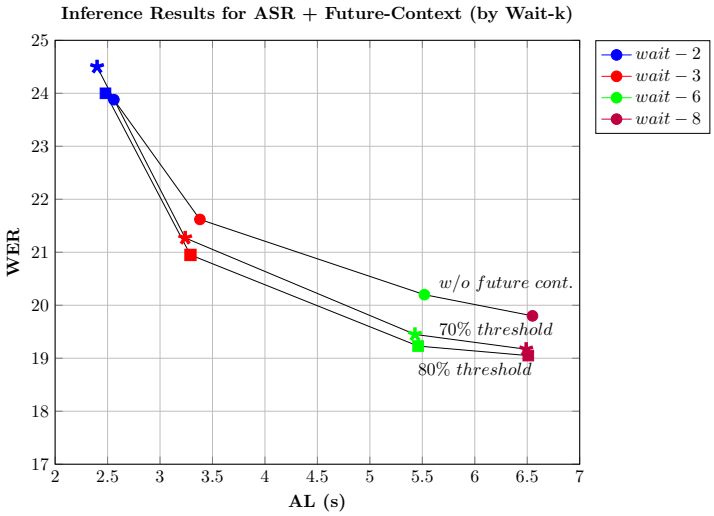
[26] Donglei Yu et al. "Self-Modifying State Modeling for Simultaneous Speech Translation". In: (2024). URL: https://aclanthology.org/2024.acl-long.528/.

[27] Xingshan Zeng, Liangyou Li, and Qun Liu. "RealTranS: End-to-End Simultaneous Speech Translation with Convolutional Weighted-Shringking Transformer". In: (2021). URL: https://aclanthology.org/2021.findings-acl.218/.

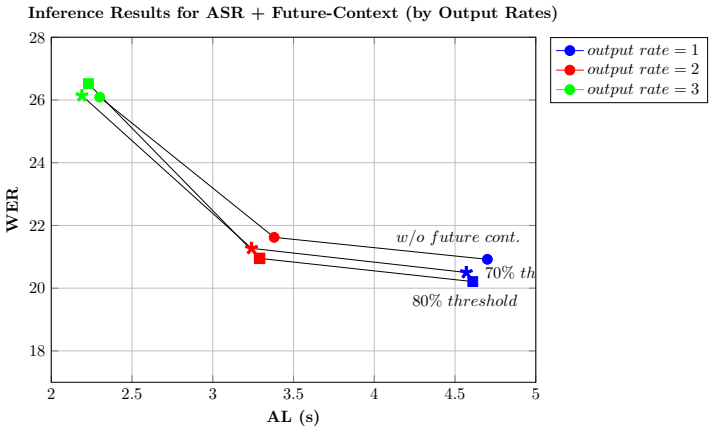
[28] Fengbin Zhu et al. *Retrieving and Reading : A Comprehensive Survey on Open-domain Question Answering*. 2021. URL: https://arxiv.org/abs/2101.00774.

A. Appendix

A.1. ASR - Wait-k Implementation With Future-Context prediction

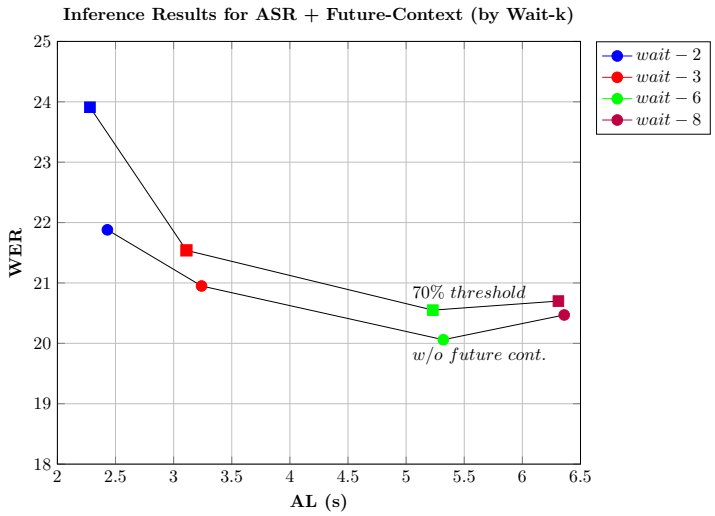


(a) Results by different wait-k values. The output rate is fixed at 2.

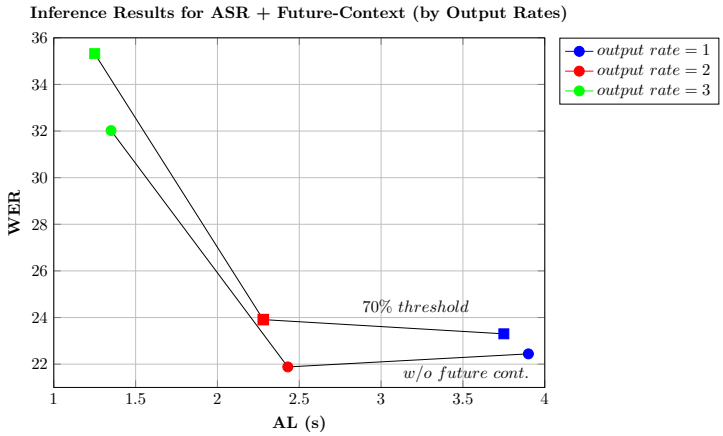


(b) Results by different output rates. The wait-k value is fixed at 3.

Figure A.1.: WER (word error rate) and AL (average lagging) results for ASR with/without token-level future-context prediction on the validation set
Future-context is predicted by the Qwen2.5-0.5B base model. Previous context size is set to 5 and only the next word is predicted.



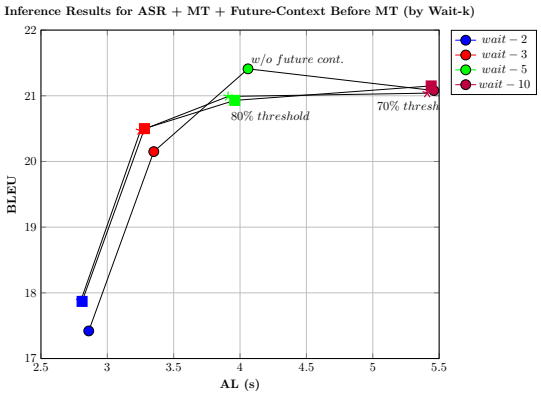
(a) Results by different wait-k values. The output rate is fixed at 2.



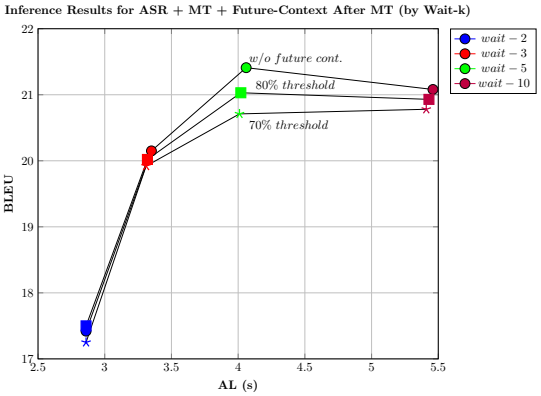
(b) Results by different output rates. The wait-k value is fixed at 2.

Figure A.2.: WER (word error rate) and AL (average lagging) results for ASR with/without token-level future-context prediction on the test set
Future-context is predicted by the Qwen2.5-0.5B base model. Previous context size is set to 5 and only the next word is predicted.

A.2. Cascaded System - Wait-k Implementation With Future-Context prediction

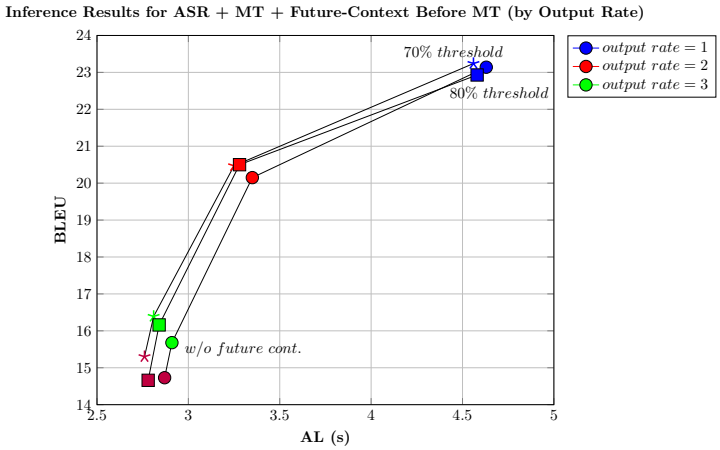


(a) Results when future-context is predicted before MT, right after each ASR step.

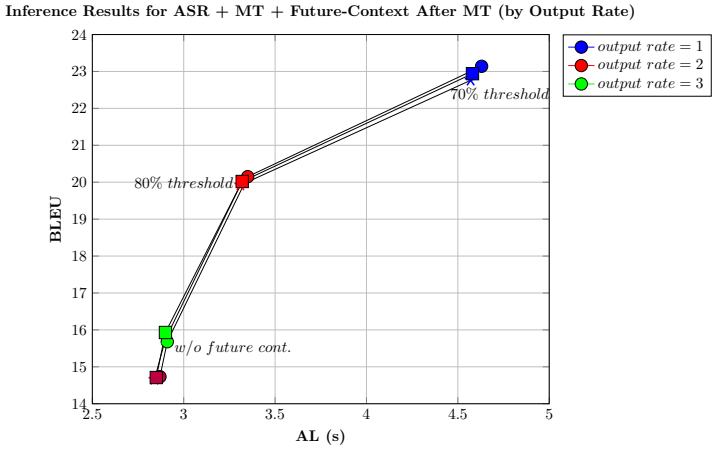


(b) Results when future-context is predicted after MT, right after each MT step.

Figure A.3.: BLEU score and AL (average lagging) results for cascaded low-latency speech translation system with/without token-level future-context prediction by different wait-k values on the validation set.
ASR wait-k strategy is same for all setups: wait-2 with output rate of 2. The output rate for the MT is fixed at 2. Previous context size is set to 5. Only the next word is predicted as future-context.



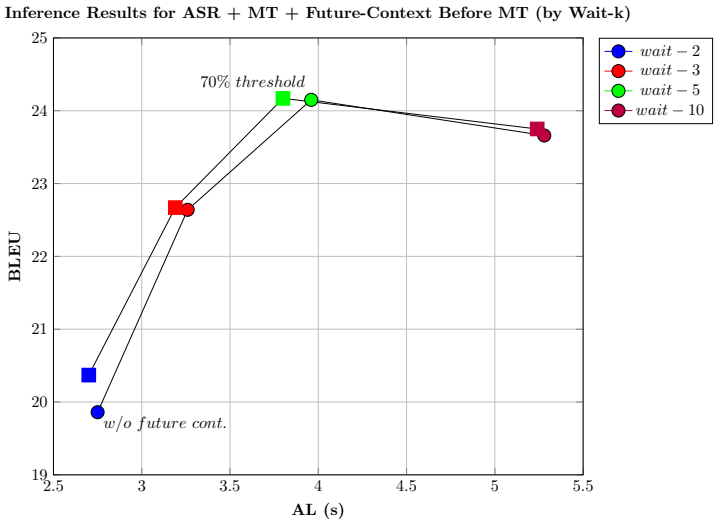
(a) Results when future-context is predicted before MT, right after each ASR step



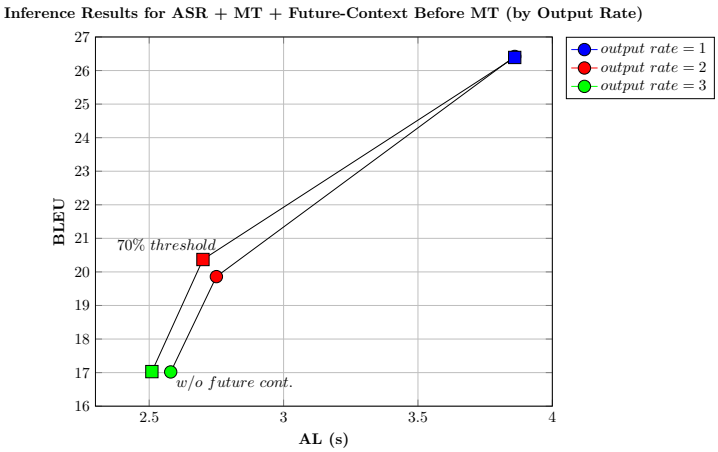
(b) Results when future-context is predicted after MT, right after each MT step

Figure A.4.: BLEU score and AL (average lagging) results for cascaded low-latency speech translation system with/without token-level future-context prediction by different output rate values on the validation set

ASR wait-k strategy is same for all setups: wait-2 with output rate of 2. The wait-k value for the MT is fixed at 3. Previous context size is set to 5. Only the next word is predicted as future-context.



(a) Results for different wait-k values. Output rate for the MT is fixed at 2.



(b) Results for different output rate values. Wait-k value for the MT is fixed at 2.

Figure A.5.: BLEU score and AL (average lagging) results for cascaded low-latency speech translation system with/without token-level future-context prediction before MT on the test set

ASR wait-k strategy is same for all setups: wait-2 with output rate of 2. Previous context size is set to 5. Only the next word is predicted as future-context.