Integrating Audio Features into Machine Translation for Dubbing

Master's Thesis of

Aren Açıkgöz

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

11. June 2023 – 11. December 2023

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, 11 December 2023

. . . . . .

(Aren Açıkgöz)

Abstract

The global surge in video entertainment consumption, commonly in English, necessitates effective solutions for rendering foreign language content accessible. This thesis focuses on automatically learning the dubbing constraints in machine translation through the integration of audio features.

The automatic learning of dubbing constraints is investigated through a rescoring method applied to a machine translation (MT) model. To build the baseline model a pretrained model DeltaLM is finetuned on OpenSubtitles and Heroes datasets. The method involves forced alignment between original audio and translation candidates, yielding scores to rescore the n-best list. The forced alignment is conducted with both Wav2Vec2 and Whisper Automatic Speech Recognition (ASR) models. This thesis also proposes a basic forced alignment decoder to Whisper and compares it with existing forced alignment modules of Wav2Vec2.

An annotated version of the Heroes dataset is the primary dataset that is used through the evaluation process. The dataset is segmented based on the visibility of the speakers. It was further observed that the method performs significantly better for the on-screen utterances than the off-screen ones. It showcases the method's ability to perform well under highly constrained conditions. Ultimately, this work aims to advance the capabilities of MT systems by integrating audio features, ensuring both linguistic authenticity and dubbing suitability.

Zusammenfassung

Der weltweite Anstieg des Konsums von Videounterhaltung, meistens auf Englisch, erfordert effektive Lösungen, um fremdsprachige Inhalte zugänglich zu machen. Diese Arbeit befasst sich mit dem automatischen Lernen von Synchronisationseinschränkungen in der maschinellen Übersetzung durch die Integration von Audiomerkmalen.

Das automatische Lernen von Synchronisationsbedingungen wird durch eine Rescoring-Methode untersucht, die auf ein maschinelles Übersetzungsmodell angewendet wird. Um das Basismodell zu erstellen, wird ein vortrainiertes Modell DeltaLM auf OpenSubtitles- und Heroes-Datensätzen Fine-tuned. Die Methode beinhaltet ein Forced Alignment zwischen dem Original-Audio und den Übersetzungskandidaten, was zu einer Rescoring einer n-Best-Liste führt. Das Forced Alignment wird sowohl mit Wav2Vec2 als auch mit Whisper Automatic Speech Recognition Modellen durchgeführt. In dieser Arbeit wird auch ein grundlegender Forced Alignment Decoder für Whisper vorgeschlagen und mit bestehenden Forced Alignment Modulen von Wav2Vec2 verglichen.

Eine annotierte Version des Heroes-Datensatzes ist der primäre Datensatz, der für den Evaluierungsprozess verwendet wird. Der Datensatz wurde anhand der Sichtbarkeit der Sprecher segmentiert. Es wurde außerdem festgestellt, dass die Methode bei den On-Screen-Äußerungen deutlich besser abschneidet als bei den Off-Screen-Äußerungen. Dies zeigt, dass die Methode auch unter stark eingeschränkten Bedingungen gute Leistungen erbringen kann. Letztendlich zielt diese Arbeit darauf ab, die Fähigkeiten von MT-Systemen durch die Integration von Audio-Merkmalen zu verbessern und sowohl die sprachliche Authentizität als auch die Eignung für die Synchronisation sicherzustellen.

Contents

List of Figures

List of Tables

1. Introduction

This chapter introduces the thesis by motivating it, discussing the research questions that the study tries to answer, and giving insight for the following chapters.

1.1. Motivation

The video entertainment industry is growing and becoming more available. According to the statistics from Statista [1] in 2022, a person spends around 19 hours on online videos on average per week. This rounds around 2.7 hours per day and if we assume an average person sleeps 8 hours per day, it is safe to say that an average person spends around 17% of their awake time on online videos. It is an incredible time commitment for an average person. A very popular streaming platform called Twitch 53.5% of the streams are in English language [16]. Among the most subscribed 50 channels on YouTube, 20 of them are uploading videos primarily in English language [2]. There are nearly 1.5 billion English-speaking people in the world [22]. It is fair to say that around 80% of the world cannot understand most of the content that is uploaded to these popular online entertainment websites.

For individuals to enjoy films and videos in foreign languages, they have the option of either watching subtitled versions or those that have been dubbed. There have been many successful researches and machine translation models [46, 23, 68, 67] to automatically generate a subtitle. Compared to dubbing subtitling is a simpler task to automate. Dubbing is generally a very costly process. Dubbing involves a multitude of factors that require careful consideration. Most importantly a good and accurate translation of the source material. The dubbed version must maintain the authenticity and emotional impact of the original one. A voice actor can represent the character and needs to keep the dubbed voice in sync with the character on-screen and talking. Otherwise, the dubbing might fail to represent the value and impact of the original film/video.

There are different approaches to dubbing. One of them is achieved by altering the videos. There are many developments and improvements in lipreading [24, 3] and lipsyncing [34, 57] which can enhance the view quality of video by manipulating the actor's lip movement to match the dubbing voice line. Another evolving method of visual manipulation is through the use of deepfakes [71, 13]. While lip-syncing primarily concentrates on adapting the text and mouth movement, deepfake techniques can alter not only a character's facial expressions but also their entire body movements within a video, or even create entirely new videos featuring the specified character. Even using deepfakes with the consent of the actors might still damage the authenticity of the original video. Such methods also require a lot of audio-visual data which will never amount to the pure-text or pure-audio data that is available. Another approach to dubbing is to adapt the machine translation(MT) models

to specifically generate translations that are fit for dubbing [41, 40, 59]. These methods manipulate or rerank the output translation texts to create a better dubbing transcript.

The general idea of the thesis is to create a method that can generate a translation for dubbing that most fit the source language audio. Specific configurations and movements of the articulators (tongue, lips, etc.) determine the phonetic characteristics of each speech sound. If the features of a speech can be extracted, they can be utilized to provide insights into the lip and tongue movements of the speaker. There have been methods [62, 36] that showed phonetic synchrony to purely text-based models. This work tries to find phonetic synchrony through the use of available audio data. This is done through extracting speech features with available Automatic Speech Recognition(ASR) models and creating a new metric based on it. Using this method, dubbed audio and mouth synchronization may be achieved without making any alterations to the original video.

1.2. Research Questions

This thesis is centered around two primary research questions (RQs), each addressing key aspects of the dubbing process:

RQ1: Can the dubbing constraints be automatically learned?

To address this question, a rescoring method will be implemented on the baseline machine translation (MT) model. By conducting a forced alignment between the original audio and possible translation candidates a score will be calculated. The score will be used for rescoring between the translation candidates from the n-best list that is generated by the baseline MT model. After using this method it will be observed if the selected translations are more suitable for dubbing.

RQ2: How does adapting to dubbing constraints affect the translation quality?

This question is of importance, as fulfilling dubbing constraints should not hinder the translation quality. It is crucial to ensure that the method is not only capable of learning dubbing constraints but can also maintain a high standard of translation quality. Balancing these two factors will be instrumental in developing an effective method for adapting to the dubbing context without sacrificing linguistic integrity.

1.3. Outline

The thesis is structured as follows. Chapter 2 explains basic concepts and techniques that are important for this thesis. Chapter 3 describes some of the related works on dubbing. In Chapter 4 the methodology is proposed and the experimental setup is detailedly described. The evaluation of the experiments is done in Chapter 5. Chapter 6 concludes the thesis by giving a brief summarization of the results.

2. Background Knowledge

The fundamental topics and needed background knowledge are explained in this chapter. In Section 2.1 the concept of machine translation is explained. The Section 2.2 refers to the models and techniques that have been developed and used in neural machine translation. Section 2.3 explains the concept of automatic speech recognition and introduces some models and methods that are used in this thesis.

2.1. Machine Translation

Machine translation(MT) is the concept of translating a language into a different language using the computer. This field of translation can be categorized into two primary approaches: Rule-based machine translation and Corpus-based machine translation.

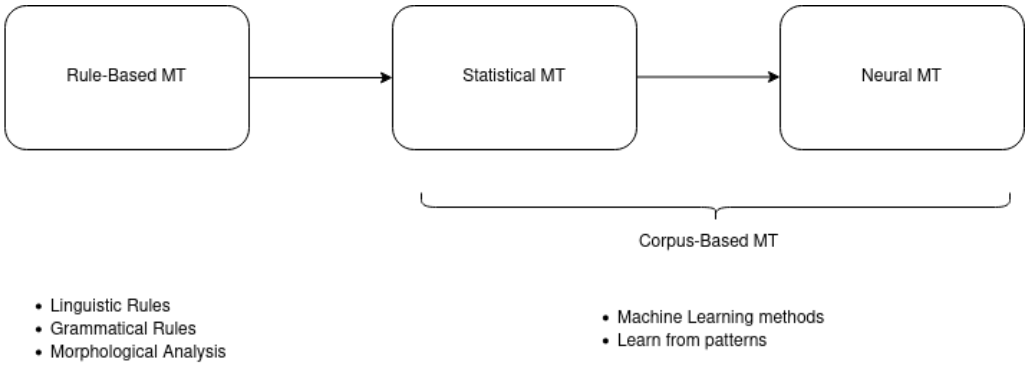


Figure 2.1.: Rule-based vs corpus-based machine translation.

2.1.1. Rule-based Machine Translation

Rule-based machine translation systems use linguistic rules and dictionaries to translate text. The translation process is defined by predefined linguistic and grammatical rules between source and target languages. Typically, the translation process initiates with a morphological analysis of the source text. Following that a bilingual dictionary is used to translate the parts of text to the target language. In the last stage, the translated parts of the target language need to be reassembled again by doing an analysis.

2.1.2. Corpus-based Machine Translation

Corpus-based approaches require huge corpora between source and target language pairs. From these corpora, a model can discover patterns between both languages and make predictions based on those findings. Corpus-based MT can be categorized into two parts: Statistical Machine Translation and Neural Machine Translation.

2.1.2.1. Statistical Machine Translation

Statistical machine translation(SMT) uses statistical methods to translate a source language text into a target language. In SMT a text is translated based on the probability distribution $p(e|f)$. Here e represents a string in the target language and f a string in the source language. There are many approaches to solving this problem. One of the most popular approaches is done by applying Bayes Theorem.

$$p(e|f) \propto p(f|e)p(e) \tag{2.1}$$

Where $p(f|e)$ is called the translation model. It gives the probability of the source string being the translation of the target string. The $p(e)$ is called the language model and it is the probability of the word e being in the target language. With the use of these subproblems, the best translation is defined as the highest probability $\tilde{e}$ that satisfies the function.

$$\tilde{e} = arg\max_{e \in e^*} p(e|f) = arg\max_{e \in e^*} p(f|e)p(e) \tag{2.2}$$

2.2. Neural Machine Translation

Neural Machine Translation (NMT) is a machine translation methodology that employs neural networks for the translation process. Neural networks became prominent during the mid-2010s [8, 35, 14, 64]. Big tech companies like Google also took an interest in this topic and made major developments [70, 66]. A major reason for that was that NMT had several advantages over SMT. With the use of neural networks, it can learn and adapt to the context of sentences rather than relying on purely statistics. NMT models can capture intricate linguistic features, and word context, resulting in more fluent and contextually accurate translations. Neural networks having an end-to-end learning process reduces the need for manually crafted linguistic rules and bilingual dictionaries.

2.2.1. Neural Network Models

The first idea of neural networks(NN) was published in 1943 [25] and 1947 [56] by Warren McCulloch a neurophysiologist and Walter Pitts a logician. The first idea derived from finding the logical background of how a neuron worked. They hypothesized that a neuron's response to incoming impulses would be an aggregation of those impulses, shaping the foundational principles of neural network theory. In Figure 2.2 the McCulloch-Pitts neuron or its more popular name the perceptron can be observed. The perceptron takes a vector as an input and gives out a weighted sum of that input vector. The components of a

perceptron are a feature vector $X = (x0, x1, x2, ...)$, weight vector $W = (w0, w1, w2, ...)$, an activation function: $\sigma()$ and a bias b. Activation functions allow NNs to learn and represent complex patterns in data by introducing non-linearity to them. An output y for a perceptron is formalized as follows.

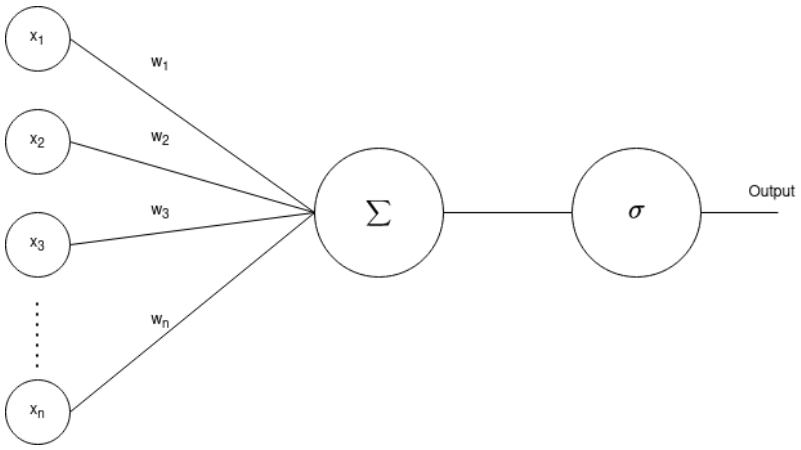


Figure 2.2.: Perceptron(McCulloch-Pitts neuron).

$$y = \sigma(\sum_{i=0}^{n} x_i w_i + b_i) \tag{2.3}$$

For tasks that are complex to represent via a single perceptron, feed-forward neural networks(FFNN) are used. FFNNs are also known as multi-layered perceptions. In FFNN each neuron has its weights and biases. FFNNs are one-directional and don't have any feedback loops. Neurons within each layer receive input from neurons from the previous layer and transmit their output to neurons in the subsequent layer, facilitating the structured flow of information. FFNNs are used for a wide array of tasks but they are not suited for tasks like machine translation where the sequence of inputs holds importance for the context of the task. Therefore sequence-to-sequence models are used.

2.2.1.1. Sequence-to-Sequence Models

Sequence-to-sequence (Seq2Seq) models are specifically made for tasks involving sequences of variable lengths as inputs. Tasks such as MT where the input is not of the same length can change in accordance to the length of sentences. Seq2Seq models are typical of encoder-decoder architecture. As the name suggests encoder-decoder models consist of two components. The first component is the encoder which summarizes input data of non-fixed length and sends it to the second component decoder. The decoder takes the output of the encoder and returns an output sequence at the end. For the encoder and decoder parts of Seq2Seq models, generally, Recurrent Neural Networks or Long Short-term Memory Networks are used.

2.2.1.2. Recurrent Neural Networks

Recurrent Neural Networks (RNN) can process sequential data and maintain the memory of previous hidden states by keeping a recurrent connection between the hidden states. RNN's capability of having somewhat of a preservation to previous states makes it a great candidate for natural language processing (NLP) tasks such as MT and ASR. In Figure 2.3 we can see an RNN. In the RNN the intermediate hidden layers are connected to their matching inputs and the hidden layer before it. The last hidden layer of the encoder encapsulates all the information of all previous layers in a so-called memory vector. The encoder then passes this memory vector to the decoder. Let's think of a machine translation task and go through the steps of an auto-regressive decoder. The decoder makes its first prediction on which word the new sentence starts. The predicted word is then fed to the input of the next decoder layer whilst having a connection to the previously hidden layer of the decoder. The decoder continues this process till the end-of-sentence token is generated or the sequence length is reached.

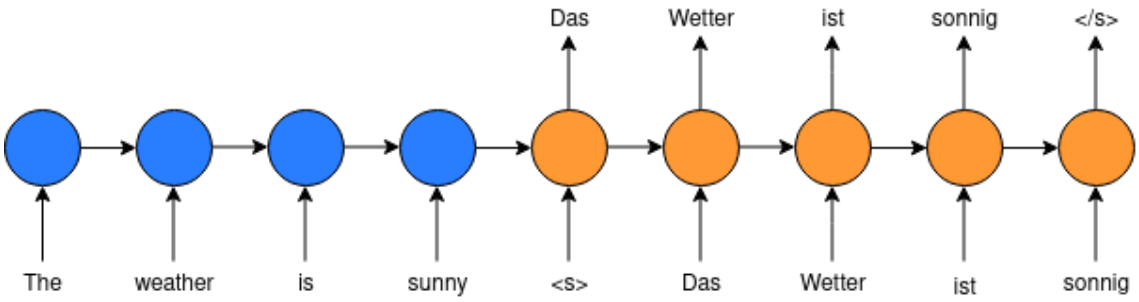


Figure 2.3.: An example of an autoregressive recurrent neural network with encoder decoder-based architecture. The encoder is the blue part and the orange neurons are the decoder parts.

It is undeniable that NLP tasks can be done by RNN yet in the current state-of-the-art models RNNs are rarely used. The main issue that RNNs suffer from is the vanishing gradient problem. During the training of the model, the gradients can become very small with the back-propagation through time steps. The small gradients then have little to no effect on the weights of the hidden states. Which slows or eventually stops the training process of the affected neurons. In the end, it is very difficult for the model to learn the long-term dependencies.

2.2.1.3. Long Short-Term Memory

Long Short-Term Memory (LSTM) [30] is an upgraded version of the RNN architecture designed to address the shortcomings of traditional RNNs. Mainly focuses on the vanishing gradient problem and RNNS's inability to capture long-term dependencies. LSTMs employ a more complicated cell structure in their hidden states. The flow and preservation of the information are controlled with memory cells and gates. These gates enable LSTMs to selectively update and forget information from previous time steps. This allows the LSTMs to learn long-distance dependencies better than a traditional RNN.

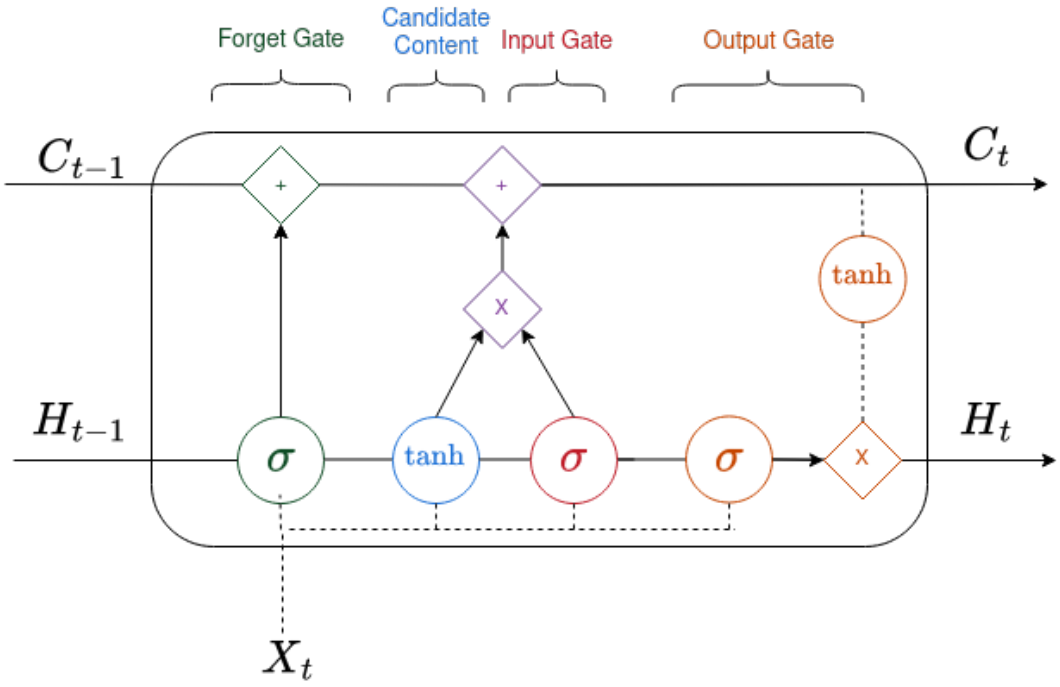


Figure 2.4.: LSTM cell.

In Figure 2.4 a close-up look at an LSTM cell is given. There are four gates in an LSTM cell. The first gate is the forget gate. The model selects and removes the information it deems unnecessary from the previous states. With the current input X_t and the previous memory H_{t-1} the forget gate is defined as $F_t = \sigma(W^{FX}X_t + W^{FH}H_{t-1} + b^F)$. Next input gate and candidate content generation create a new cell content C_t. The input gate: $I_t = \sigma(W^{IX}X_t + W^{IH}H_{t-1} + b^I)$ and the candidate content: $\tilde{C}_t = \tanh(W^{CX}X_t + W^{CH}H_{t-1} + b^C)$ is then combined with forget gate to create the new cell content: $C_t = F_tC_{t-1} + I_t\tilde{C}_t$. After the current cell state C_t is calculated the new H_t can now be calculated with the output gate: $O_t = \sigma(W^{OX}X_t + W^{OH}H_{t-1} + b^O)$. The current state $H_t = O_tC_t$ is the combination of the output gate O_t and the current cell state C_t.

2.2.1.4. Attention-based Models

LSTMs can capture long-term dependencies, but they can still suffer from vanishing gradient problems like RNNs for very long sequences. The idea is to create a component, that checks for the context information for each output rather than cramming all the input that the encoder does to a single memory vector. A very basic idea of including this is done by adding an attention module between the encoder-decoder structure. At each decoder state, the decoder generates a query according to its state. The encoder has the key and values for each of its inputs to represent their state. When the attention module receives a query from the decoder it calculates a context vector based on the key and values the encoder provides.

2.2.1.5. Self-Attention

The paper "Attention is all you need" [66] takes this concept a little further by proposing self-attention mechanisms and a new NMT model architecture called transformer. The process of self-attention begins with an initial linear transformation. From each input vector, three different vectors are calculated. The input is transformed into query and key vectors of dimension d_k and value vector of dimension d_v. To optimize the calculations the query, key, and value vectors are packed into matrices Q, K, and V. Then the attention output of the inputs can be calculated as:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.4}$$

Since the self-attention is done within each input sequence as a whole and the whole input is known, the training process can be done in parallel. The decoding process can't be done in parallel while for each newly generated output sequence, a new attention is calculated.

2.2.1.6. Multi-headed Attention

They also proposed a method called multi-headed attention. Instead of performing a single attention function, the attention process is done multiple times with different linear projections of the inputs. For each so-called attention head, each input has a different query, key, and value vector. Multi-headed attention is formulated as:

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O \tag{2.5}$$

$$where head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{2.6}$$

2.2.1.7. Transformer

The transformer is the current state-of-the-art model that is used in MT problems. It utilizes the multi-headed self-attention technique to capture intricate linguistic dependencies and contextual relationships in source and target languages. By employing these techniques it can process entire input sequences in parallel leading to massive gains in computational efficiency. The proposed transformer architecture of "Attention is all you need" [66] can be seen in Figure 2.5.

The transformer is based on two main modules, the left module is the encoder and the right module is the decoder. On the encoder, the input embeddings go through a positional encoding section. The model does not use the order of the sequential data. To feed the model the order of the sequence, the positional embeddings are used. The positional embeddings are sin and cosine functions of different frequencies based on the position of the word and the dimension of the model. The functions are defined as:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}}) \tag{2.7}$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}}) \tag{2.8}$$

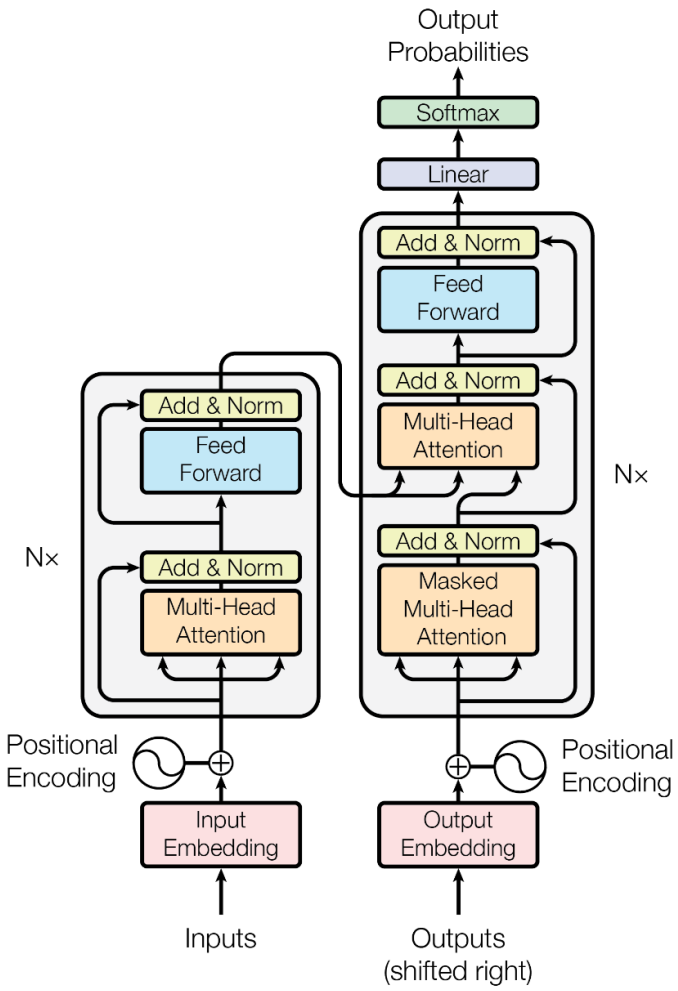


Figure 2.5.: The Transformer - model architecture[66].

where the *pos* is the position of the word and i is the dimension of the model. After the positional encoding, the first layer is a multi-headed self-attention layer. It is then followed by a position-wise fully connected feed-forward network. Both of the layers are followed by layer normalization.

The decoder consists of three sub-layers. Similar to the encoder the output embeddings are first injected with positional encoding. The first layer is a masked multi-headed attention layer. It is masked unlike the self-attention layer in the encoder because during decoding words are generated sequentially and the attention should be calculated even if the sequence is not fully finished. The second layer is a multi-headed cross-attention. The key and values are taken from the output of the encoder and the queries are from the previous layer of self-attention in the decoder. The last layer is a fully connected feed-forward network and all sub-layers are followed by layer normalization.

2.2.2. Training

The training process is a crucial part of the machine translation process. The very first step to start training a model is to find the right data for it to train. If the model is being trained on supervised learning, labeled data is needed. For MT systems labeled data is generally found as matching source and target language sentence pairs. It is not always easy to find such data that fits your system. For those times unsupervised learning is used. An example of it would be the masked language modeling technique used in BERT [18]. The idea behind that technique is to mask some words or tokens from the input sentence and as an objective to predict those masked words from the given sentences. This way the model learns the connection between the words and further allows BERT to generate deeply contextualized word embeddings, encoding nuanced semantic relationships within a sentence. Since the method does not need any parallel references, it makes good use of monolingual data.

For this thesis supervised learning is used. Therefore the labeled data at hand undergoes preprocessing to ensure that the data is clean and consistent. First, the data is separated into sections. The most common is to have a training, validation, and test set. Only the training data is used to train the model. Then in the preprocessing, the data generally is subjected to tokenization, lower-casing, and removal of some special characters. After preprocessing the data the first step of learning can start. The model processes a source sentence and generates a target sentence. This is called the forward pass. Then the system compares the generated sentence to a reference sentence and calculates the differences using a loss function. The system then tries to minimize the loss function by updating the weights of the network. The weights are updated by using iterative methods like stochastic gradient descent or versions of it like Adam [37] and Adagrad [21]. This process is repeated till some time requirements are met, then the model is tested on the validation set to see if the model can reach the expectations. This process is continued till the expectations are met.

2.2.3. Decoding

Decoding is the step where the translation is generated. In decoding intention is to select the most probable translation based on the model:

$$y^* = argmax_y P(y|x) \tag{2.9}$$

For each word in the decoding step, the decoder selects a word from the vocabulary of words based on the search and selection algorithm. One of them is greedy search in which for each step of the decoding sequence the word with the highest probability is selected. This algorithm is very lightweight but not very accurate for most situations since the combination of the most probable words might not always result in the most probable sentence. The second search mostly focuses on the sentence rather than the words. An exact search tries all word combinations given and finds the sentence which is the most probable. This is very heavy on computational and memory. In real-life applications, it is only feasible for very short sentences. Another method is sampling. A random word is

selected between the possible words. These results are very unreliable so a variation of it is mostly used in which the sampling is done between the k most probable words.

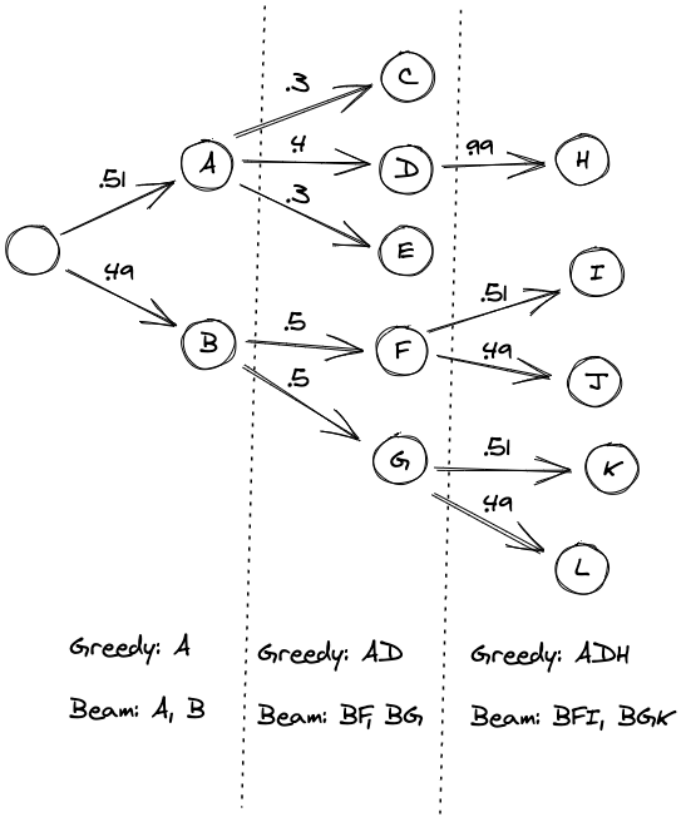


Figure 2.6.: A comparison of greedy and beam search.[1]

The last method to talk about is called the beam search. It is employed to find the most likely sequence of output words in a probabilistic model while considering multiple alternatives. A beam size is selected beforehand and the most probable n sequences are saved to the beam. In every step, the sequences in the beam are extended with new words from the decoding step. Then the beam is pruned to only have the n most probable among those extensions. For beam size zero the search algorithm is equal to greedy search and for unlimited beam size it is equal to exact search. It is feasible to think of beam size as a weight of selection between greedy and exact search.

2.2.4. Evaluation

The evaluation step is done to assess the quality of translation and MT systems. The best way to evaluate a translation is done by humans. Even though it is the best way to evaluate it is very time-consuming and expensive. Automatic evaluations are on the other hand very cheap, fast, and use human-made references. The main premise is to check the similarity between the human reference and the generated translation.

[1] https://discuss.huggingface.co/t/is-beam-search-always-better-than-greedy-search/2943

One of the most used automatic evaluation metrics is called sacreBleu [47] which derives from the original BLEU score concept [54]. It utilizes n-gram overlaps to measure the similarity between sentences. Bleu score considers one to four-gram overlaps BLEU score is calculated as $BleuScore = \sqrt[4]{P_1 P_2 P_3 P_4 BP}$. P_{1-4} denotes the n-gram overlaps of one to four and BP denotes the so called brevity penalty. For sentences where the candidate sentence is shorter than the reference sentence, this penalty is applied. A short candidate sentence can be a direct n-gram of the reference sentence in which the BLEU score results high even though it is missing part of the sentence. This is punished by the brevity penalty. It is defined as:

$$BrevityPenalty = \begin{cases} 1, & r > c, \\ e^{1-r/c}, & r \leq c. \end{cases} \tag{2.10}$$

Following is an example of a BLEU score calculation where the $BP = 1$:

Reference: I must run to the shop
Hypothesis: I can run to the shop

1-gram= $\frac{5}{6}$; 2-gram= $\frac{3}{5}$; 3-gram= $\frac{2}{4}$; 4-gram= $\frac{1}{3}$
BleuScore = $\sqrt[4]{5/6 * 3/5 * 2/4 * 1/3} = 0.53728496591$

The Bleu score is an intuitive and extensively adopted metric for evaluating the quality of machine-generated translations. Nonetheless, this method comes with certain limitations. Bleu score relies on n-gram matching, deeming translations as "good" when they exhibit exact word matches with the reference sentence. Other translations with better meaning encapsulation where there are fewer matching n-grams might be lowly rated. Another problem that derives from n-gram matching is that each matching word is treated equally. Not every word in a sentence contributes to the meaning of the sentence equally therefore this can cause problems regarding reliance to the context or meaning of candidates that are highly scored.

A metric that tries to resolve these issues is COMET [60]. COMET is a neural model to predict scores. Unlike BLEU, which primarily relies on n-gram matches without considering semantic meaning, COMET employs methods to assess content and fluency. It uses a sentence embedding model to capture semantic similarities, providing a better understanding of the translation quality by considering contextual and syntactic aspects. This way COMETs evaluation can be closer to a human evaluation process, especially in cases where BLEU might fall short due to its focus on exact word matches.

2.3. Automatic Speech Recognition

Automatic Speech Recognition(ASR) is the method for understanding spoken language and transcribing it via computers. State-of-the-art solutions mostly employ machine learning techniques. A traditional structure of an ASR system can be observed in Figure 2.7. The audio is first captured and preprocessed by removing the noise. Then the clean audio data

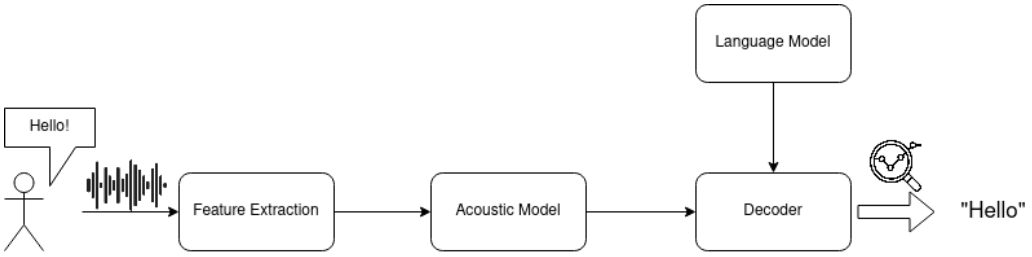


Figure 2.7.: A traditional automatic speech recognition model structure.

goes through feature extraction. In this step, the audio data is transformed to suit the input requirements of a machine learning model, ensuring that relevant components are preserved for subsequent analysis. The extracted data is given to an acoustic model which is generally a deep-learning model which recognizes phonemes and acoustic patterns. The patterns were then matched to possible word sequences with a language model. Methods like Connectionist Temporal Classification [27] are used to align these audio frames with linguistic units. Most ASR systems [58, 72] further utilize a re-casing and punctuation model to refine the output transcript.

This study uses state-of-the-art ASR models to understand the audio features. Therefore detailed knowledge of these models is useful to grasp the concepts in this thesis. The following two Sections 2.3.1 and 2.3.2 explain the methods and capabilities of the models Wav2Vec2 and Whisper.

2.3.1. Wav2Vec 2.0: A Framework for Self-Supervised Learning of Speech Representations

This section will summarize the ASR model wav2vec2. Wav2vec2 is used in this thesis and plays a crucial role in understanding the features of given audio files.

Wav2Vec 2.0 (Wave-to-Vector 2.0) [7] is a state-of-the-art automatic speech recognition model developed by Facebook AI[2]. It is the successor of Wav2vec [63]. The primary motivation driving the authors is the scarcity of extensive sets of labeled data. Building effective speech recognition models necessitates access to thousands of hours of labeled speech data, a resource that is challenging to obtain, even for widely spoken languages. The situation becomes even more daunting when considering low-resource languages, where the prospect of amassing such volumes of data is, in practical terms, beyond reach.

To lower the necessity of high amounts of labeled training data authors of [7] propose a framework for self-supervised learning using raw speech data. This approach involves the utilization of convolutional neural networks (CNNs) [50] to process unannotated speech data, and subsequently applies masking techniques to derive latent speech representations, drawing from prior research such as [33] and [69]. Then the mask data is sent to a transformer network to be contextualized. Similar methods have been done before [33, 6]

[2]Facebook AI: ai.meta.com

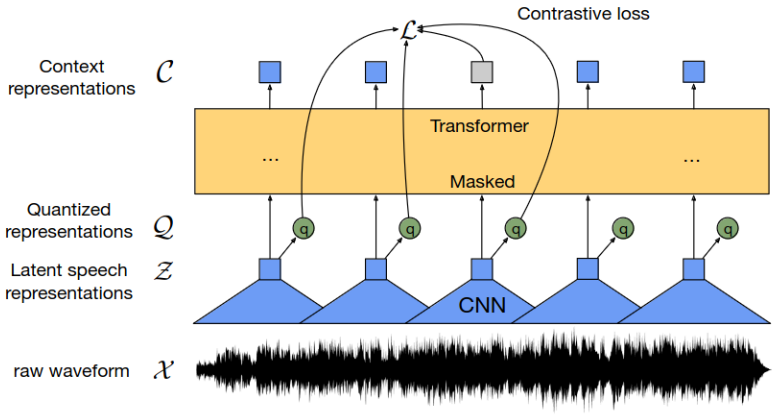


Figure 2.8.: Wav2vec2 framework learning contextualized speech representations and discretized speech units [7].

in cascaded architectures, the authors employ the method in an End-to-end way. After the first part of the training, the model is fine-tuned with labeled data using CTC [27] loss.

2.3.1.1. Model

The framework and the model of Wav2Vec2 can be observed in Figure 2.8. The model consists of three main components. The feature encoder is built on CNN layers followed by layer normalizations [5]. The second component is the Transformer [66]. The transformer takes the output of the feature encoder and learns the contextual representations. The positional embeddings are included as relative positional embeddings with convolutional layers [6, 48] instead of fixed positional embeddings. The last component is the quantization module. The model learns discrete speech units [**discrete units**, 43, 15] by using a Gumbel softmax [31] to represent the latent representations.

2.3.1.2. Datasets

For different experimental setups, the model is trained on a variety of datasets. For the unlabeled data, two different datasets are chosen. A larger dataset consisting of 960 hours of unlabeled speech data from Librispeech corpus [53] and an around 60k hour audio dataset from LibriVox[3]. For fine-tuning five labeled dataset settings are used. 960 hours of labeled speech from Librispeech, 100 hours from the train-clean-100 subset, and Librispeech subsets consisting of 10 hours, 1 hour, and 10 minutes of labeled data. For validation and testing the dev and test sets of Librispeech are used.

2.3.1.3. Training

The training of the model is separated into two sections. The pre-training is where the model is trained with unlabeled data and fine-tuning is where the model is fed labeled data.

[3]https://librivox.org/

In the pre-training, the unlabeled data is converted to latent speech representations and then masked before being fed into the transformer. The inputs that go into the quantization module are not masked. Masked sections are selected through sampling from all time steps to be starting indices of the mask and from that point 10 subsequent time steps are masked. This equates to masking 49% of all time steps. The objective loss function of the pre-training is an aggregation of a contrastive task L_m forcing the model to make correct differentiations between the true quantized latent speech representation for the masked time intervals and a diversity loss L_d encourages the model to make use of codebook entries more often. The loss function is defined as follows where α is a hyperparameter:

$$L = L_m + \alpha L_d \tag{2.11}$$

After the pre-training, the model is fine-tuned on labeled data. During fine-tuning the feature encoder is not trained.

2.3.1.4. Decoding

In decoding, two language models(LM) are used. The first LM is a 4-gram model and the second one is a Transformer based LM trained on Librispeech LM corpus. The search strategy that is used is beam search which the beam size varies for each LM. For the 4-gram model the beam size is selected as 1500 and for transformer LM it is 500.

2.3.2. Whisper: Robust Speech Recognition via Large-Scale Weak Supervision

The authors of [58] propose a zero-shot ASR system that differs from the other state-of-the-art approaches like [7] by using weakly supervised learning. As mentioned in Section 2.3.1 Wav2Vec2 uses unsupervised learning to train its model with the use of unlabeled data. The authors of Whisper[4] strongly point out that training with high amounts of unlabeled data and low amount of labeled data causes the model to have an imbalance of quality between its encoder and decoder. The encoders are trained with high amounts of unlabeled data and therefore can learn high-quality speech representations. On the other hand, the decoder is mostly trained in the fine-tuning stage with low amounts of labeled data. Because of the labeled data scarcity, the decoders can only perform well in the context of labeled data. To address this problem supervised training is used in Whisper.

2.3.2.1. Data Processing

More labeled data would increase the quality of the model. The reason behind using self-supervised learning is to compensate for the lack of quality labeled data with the use of easily obtainable unlabeled speech data.

The motivation of the authors is derived from the developments in the recent work in computer vision. In computer vision, the use of crowd-sourced large but weakly supervised datasets like ImageNet [61] had great improvements upon the models [38]. Even though these datasets are larger than the existing higher-quality labeled ones they are very small

[4]WSPSR standing for Web-scale Supervised Pretraining for Speech Recognition

compared to the unlabeled ones. For whisper authors collect 680,000 hours of labeled audio data. 117,000 hours of that data including 96 different languages. The dataset also contains 125,000 hours of translation data from various languages to English.

The dataset is assembled by audio files that are paired with their transcriptions from the internet. The process involves gathering a wide array of audio samples from various sources, including podcasts, YouTube videos, and other spoken content available on the internet. Through this method, the dataset contains content from a lot of different contexts and becomes diverse. Learning from various qualities of audio can increase the performance against noisy data and the robustness of the model. The same cannot be said about the varying quality of transcriptions, it can blur the mind of the model. To prevent this authors employ automatic filtering methods. They found that many of the transcripts online are generated through other ASR systems and mixing human and machine-generated data can impair the systems performance [26]. Therefore they use methods that detect if a transcript is human- or machine-generated. They check if the transcript is well punctuated and correctly cased. As for other filtering methods, they use a language detector and check if the given transcript matches the audio file.

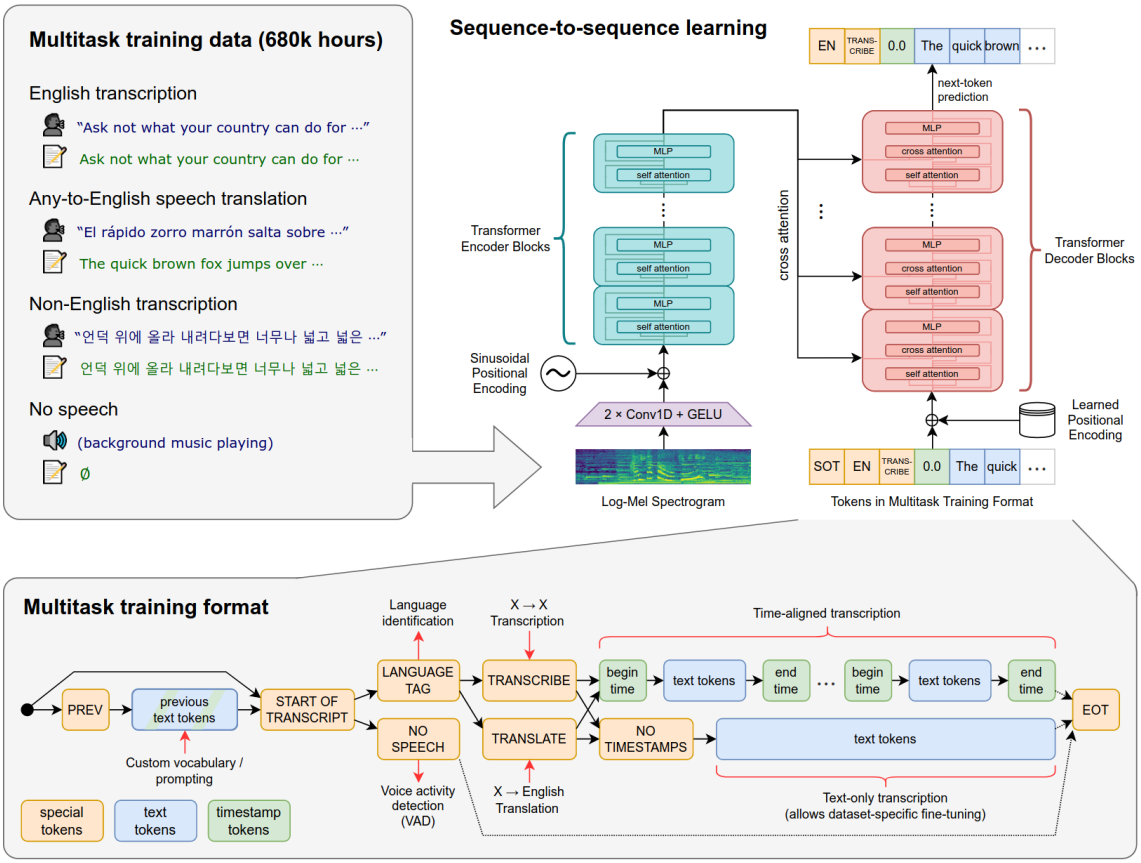


Figure 2.9.: An overview of Whisper's model and decoding structure [58].

2.3.2.2. Model

Whisper focuses on showing the competitiveness of the weak supervised learning method. They use a basic transformer model similar to the speech transformer [20] which is a modified transformer [66]. In the upper right of Figure 2.9 an overview of the model can be seen. To process audio data the transformer architecture is extended by two convolution layers and the GELU activation function [28]. The embeddings then go through a sinusoidal positional encoding and then inputted into the transformer's encoder blocks. The encoders and decoders are the same as the original transformers. The amount of encoder blocks is equal to the amount of decoder blocks.

2.3.2.3. Training

Five different models are trained to observe the differences in model sizes. Different model sizes can be seen in Figure 2.10. The models are trained with AdamW [44] and a linear learning rate decays to zero after the first 2048 warm-up steps. They are trained for 2-3 epochs, therefore there were no precautions taken for over-fitting.

Model	Layers	Width	Heads	Parameters
Tiny	4	384	6	39M
Base	6	512	8	74M
Small	12	768	12	244M
Medium	24	1024	16	769M
Large	32	1280	20	1550M

Figure 2.10.: The architectural differences of whisper models [58].

2.3.2.4. Multitask Format

The principal function of an Automatic Speech Recognition (ASR) system resides in the transcription of audio files. However, its utility extends beyond mere transcription. Whisper is capable of completing other tasks such as speaker diarization, voice activity detection, language detection, inverse text normalization, and even in unidirectional $x \rightarrow$ "*en*" translation. To keep the system clean and straightforward, all the tasks are packed in a single processing pipeline. The pipeline can be observed in the lower part of Figure 2.9.

To specify the beginning and end of a transcription `<|startoftranscript|>` and `<|endoftranscript|>` tokens are used. First, the language of the given speech segment is predicted. If no speech is detected the `<|nospeech|>` token is appended to the current tokens and then the transcription is finalized. Upon language detection, the subsequent step necessitates task specification. Task differentiation within this unified pipeline is achieved through the utilization of task-specific tokens. `<|transcribe|>` or `<|translate|>` token is appended and following that a `<|notimestamps|>` token

is used, whether to predict timestamps or not is specified by the user. When the task definition is finalized the model starts predicting the audio segments from its vocabulary. The audio segments are quantized to the nearest 20ms which is the time resolution of the whisper. After the transcription of the final audio segment `<|endoftranscript|>` token is appended and the tokens are truncated and sent to the tokenizer. The tokenizer can encode text into tokens or decode tokens into text. Then the output is assembled and the process is finalized.

Dataset size	English WER (↓)	Multilingual WER (↓)	X→En BLEU (↑)
3405	30.5	92.4	0.2
6811	19.6	72.7	1.7
13621	14.4	56.6	7.9
27243	12.3	45.0	13.9
54486	10.9	36.4	19.2
681070	**9.9**	**29.2**	**24.8**

(a) The effect of data amount to the performance of the model.

Dataset	wav2vec 2.0 Large (no LM)	Whisper Large V2	RER (%)
LibriSpeech Clean	**2.7**	**2.7**	0.0
Artie	24.5	**6.2**	74.7
Common Voice	29.9	**9.0**	69.9
Fleurs En	14.6	**4.4**	69.9
Tedlium	10.5	**4.0**	61.9
CHiME6	65.8	**25.5**	61.2
VoxPopuli En	17.9	**7.3**	59.2
CORAAL	35.6	**16.2**	54.5
AMI IHM	37.0	**16.9**	54.3
Switchboard	28.3	**13.8**	51.2
CallHome	34.8	**17.6**	49.4
WSJ	7.7	**3.9**	49.4
AMI SDM1	67.6	**36.4**	46.2
LibriSpeech Other	6.2	**5.2**	16.1
Average	29.3	**12.8**	55.2

(b) The comparison between Wav2Vec2 and Whisper on various datasets.

Figure 2.11.: Whispers performance[58].

2.3.2.5. Evaluation

Whisper has showcased that the use of weak supervised learning had improvements on unsupervised models. In Figures 2.11 the positive influence of weak supervision can be observed. In the table to the left, the impact of data amount can be observed. With increasing data size the word error rate(WER) decreases substantially for both English and multilingual cases. Improvements in unidirectional $X \rightarrow$ "*en*" translation can also be observed. The table to the right indicates a comparison between Wav2Vec2 and Whisper in different test datasets. Since Wav2Vec2 is a state-of-the-art model based on unsupervised learning, differences between the learning methods can be observed. The comparisons are made in a zero-shot environment in which none of the models are further fine-tuned to fit the test dataset context. It is well known that Wav2Vec2 requires fine-tuning for specific task usage and its performance in zero-shot scenarios is therefore limited. Whispers major superiority can be observed for all the datasets but the Librispeech-clean set. The reason for this is that the Wav2Vec2 model is fine-tuned on Librispeech. Even in the scenario where Wav2Vec2 is fine-tuned, the Whisper model still stays competitive. Showcasing that without requiring fine-tuning or user expertise Whisper can perform as well as a fine-tuned Wav2Vec2 model.

2.3.3. Forced Alignment

As stated in the previous Section 2.3 ASR systems take in an audio sequence and in return output a transcript matching that sequence. Within decoding of the audio sequence, the language model tends to return the most probable text sequence. Observing that probability, the confidence of the system can be measured per given sequences. Forced alignment is analyzing the acoustic features of the audio and matching them to the linguistic units in the provided text. In Figure 2.12, two graphs can be observed from which the alignment probabilities and time segments of the sentence "I had that curiosity beside me at this moment". The first graph shows a path with labels and the probabilities of each linguistic unit matching with acoustic features. The second graph showcases each label's probability. Forced alignment is particularly valuable in scenarios where precise synchronization between spoken words and their textual representations is essential.

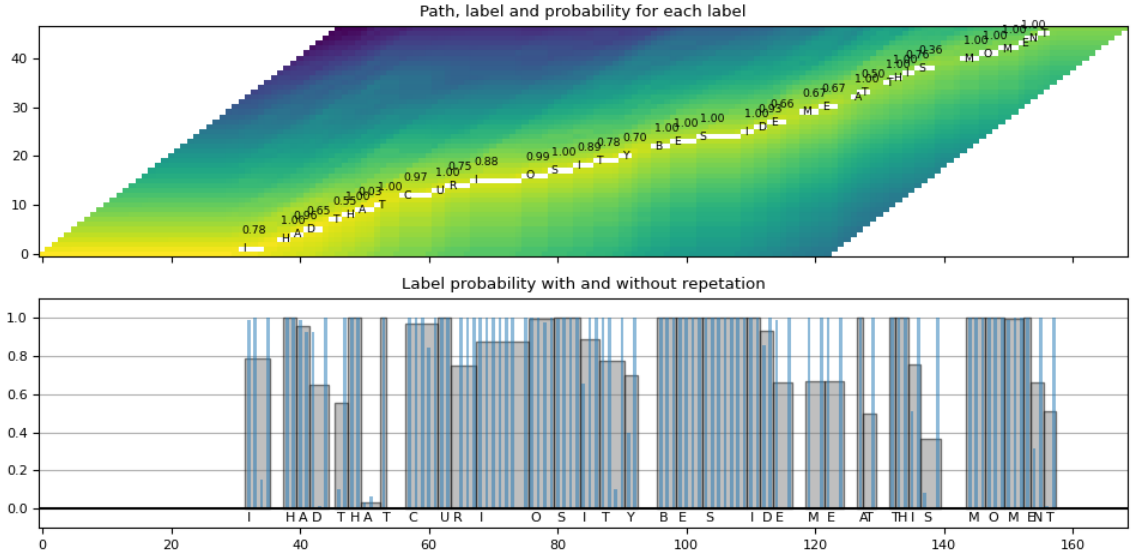


Figure 2.12.: A forced alignment example. The upper graph is a representation of each word's alignment with the time segments of the audio. The bottom graph shows the probability of each word for the given time-segment[29].

3. Related Work

This chapter discusses and explains important state-of-the-art works, that are used throughout this thesis. Section 3.1 introduces an improved Whisper model, in Section 3.2 the dubbing constraints are discussed, Sections 3.3 and 3.4 showcases a rescoring method that have been used in the evaluation of this thesis.

3.1. Whisperx: Time-Accurate Speech Transcription of Long-Form Audio

The authors of [9] propose a framework Whisperx in which they showcase a method to do time-accurate speech transcriptions of long audio sequences. They emphasize that state-of-the-art models like [58, 7] tend to segment the audio in certain sections like 30 seconds (Whisper) and then transcribe that segment. This can certainly prohibit the transcription quality. Specifically, if segmentation occurs mid-sentence, the model may not properly capture the entirety of the sentence's contextual information during training. Moreover, if the segmentation happens in the middle of a phoneme a word might get lost. The authors propose new 3 steps in addition to the current transcription stages of Whisper. First pre-segmenting the input audio. Secondly cutting and merging the segmented audio bits to fit the requirements of Whisper. Finally, the third stage is a forced alignment procedure, leveraging an external phoneme model.

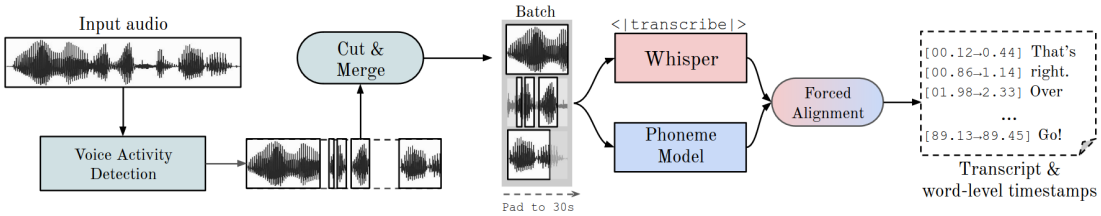


Figure 3.1.: WhisperX: A general view of the pipeline. Firstly it starts with voice activity detection(VAD) then a module called "cut and merge" where parts are adjusted to fit the ASR model's requirements and lastly, the forced alignment module employs 2 different ASR models. Whisper for transcription and another one for aligning [9].

Voice Activity Detection

Voice activity detection (VAD) is the process of finding the speech-active regions of the audio stream and segmenting them into regions. By doing this pre-segmentation

unnecessary forward passes can be mitigated by eliminating the no-speech zones of the audio stream. The sentences that continue through the boundaries of audio chunks can now be properly segmented and as a bonus the audio word-level segmentation can be achieved easily.

VAD is a sequence labeling task where the input audio is $A = a_1, a_2, ..., a_T$ represented as a sequence of acoustic features and the output $y = y_1, y_2, ...y_T$ are binary sequence labels of the audio. For further use, the binary sequences can be seen as active speech segments $s = s_1, s_2, ..., s_N$, where the $s_i = (t_0^i, t_1^i)$ start and end indexes of the speech segment is given.

Cut & Merge

In the last Section 3.1 the segmentation of the audio is done and no active speech zones are partially separated. The audio stream is now fragmented and contains a lot of small audio segments. This poses a challenge, as each of these smaller segments necessitates an individual forward pass through the Whisper model which unveils a substantial amount of computational overhead that reduces the efficiency and speed of the process. On the other hand, there will be some segments that are longer than the maximum input duration of the Whisper, therefore they cannot be transcribed in a single forward pass. As a solution, the authors propose a "min-cut" operation to provide an upper bound to these audio segments.

To guarantee that the segments are not short and also not longer than the requirements of Whisper, the length of audio-segments is constrained between $\frac{1}{2}|A_{train}|$ to $|A_{train}|$. Where $|A_{train}|$ is the maximum input audio duration of Whisper. The longer segments are cut and the shorter segments are merged to fit these restrictions. The merge operation is done by merging neighboring segments where the aggregation of the segments is shorter than $|A_{train}|$.

Transcription

The transcription is done with the ASR model Whisper. Last Aection 3.1 provided optimal input speech segments for Whisper. With the provided speech segments can be transcribed in parallel for extra efficiency. Performing the transcription in parallel concretizes the independence of each segment, not letting the text condition itself upon the previous segment.

Word-level Forced Alignment

Lastly, WhisperX introduces a forced alignment module. To find each phoneme in the audio sequence a phoneme recognition model is used, which is trained to classify and separate each word in the audio. For input audio segment S, the model outputs a logit matrix $L \in \mathbb{R}^{KxT}$ where K is the number of phoneme classes and T varies on the resolution of the model. For each audio segment S and text pair $\mathcal{T}_i$, first a set of phoneme classes $C_{\mathcal{T}_i}$ are defined by using the segmented corresponding text. the phoneme classification is employed upon the segment with the classes $C_{\mathcal{T}_i}$. Dynamic Time Warping(DTW) is employed on the logits matrix L_i to find the optimal path of the phonemes in the given text $\mathcal{T}_i$. After that, the start and end time of each word can be accessed through the phonemes.

Using VAD and Cut & Merge for pre-processing methods, WhisperX achieves a 12-times speed up against Whisper through parallelization of the inputs. In addition to that having the segment transcriptions enables for an easy application of forced alignment through the use of a phoneme model.

3.2. Integration of Dubbing Constraints into Machine Translation

The Authors of [62] are one of the pioneers who introduced dubbing constraints into machine translation. They were motivated by the developments in speech translations [32]. However, the models had no considerations for dubbing tasks. There are important factors to consider for dubbing to achieve a professional level. This work mostly focuses on the lip-syncing element in dubbed speech. However, any model that trains on audio-visual data will lack since it will always remain smaller compared to pure text data. Any constraint therefore should be first considered for purely text-trained MT models. Also some scenes where the speech is off-screen these constraints will not be as relatively important. The paper proposes a constraint:

$$score_d^{\alpha}(C,S) = (1-\alpha) \times score_t(C,S) + \alpha \times score_p(C,S) \tag{3.1}$$

where C are the candidate utterances, S is the sentence in the source language, $score_t(C,S)$ is the translation score that the MT model determines, and $score_p(C,S)$ is a phonetic synchrony score. With a hyperparameter α, the importance of synchrony can be selected variably for different types of scenes (speech on-screen, speech off-screen).

They further define the phonetic synchrony score:

$$score_p(C,S) = \frac{1}{1 + sylldiff\,(C,S)} \tag{3.2}$$

$$sylldiff\,(C,S) = abs(syll(C) - syll(S)) \tag{3.3}$$

where $syll(x)$ is defined as the number of syllables in the sentence x. As for translations they use the BLUE score to measure the translation quality of the translation and they propose a metric called synchrony-score to measure the optimality of the translation for dubbing purposes:

$$synchrony - score(T) = \frac{\sum_{e \in T} abs(syll(NMT(e)) - syll(e))}{\sum_{e \in T} syll(e)} \tag{3.4}$$

where $NMT(e)$ is defined as the translation sentence that is generated by the MT model for the source sentence e.

3.3. The Two Shades of Dubbing in Neural Machine Translation

In "The Two Shades of Dubbing in Neural Machine Translation" [36] the authors point out that the translation done for the on-screen scenes varies from the one done for off-screen.

The on-screen scenes where the actor's mouth is visible on screen should be constrained by synchronization of the original audio and the dubbed one. However, this does not apply to the off-screen scenes where the there the speaking actor is not on-screen. Constraining a translation towards fitting the lip movements, matching the pauses and the length of the original text might cause losses in the translation quality. Having these in consideration a separation of on- and off-screen should not be treated equally in the context of dubbing.

First, the authors annotate and separate the Heroes dubbing dataset [51] into on- and off-screen sections. Then they use a transformer model for translation and compare the results.

Data, Annotation, and Analysis

Data as specified before is the Heroes dataset. On this dataset, two annotators worked and determined what was on and off the screen. One of the annotators watches all episodes and identifies the scenes where the mouth of the character is visible or not. Each utterance can be "on", "off" or mixed. Mixed means that the mouth is visible only partially for an utterance. A second annotator works on 10% of the dataset for validation. The annotated set is publicly available online[1].

Ratio	ON	OFF	MIXED
Character	1.10 (0.27)	1.10 (0.38)	1.13 (0.29)
Syllable	0.90 (0.26)	0.90 (0.36)	0.93 (0.27)
Vowel	0.92 (0.28)	0.93 (0.38)	0.94 (0.29)
Consonant	1.27 (0.41)	1.28 (0.56)	1.27 (0.39)
Duration	1.04 (0.31)	1.07 (0.36)	1.04 (0.24)

Figure 3.2.: For each section the mean and standard deviation for the source and translated sentences average ratio of different units [36].

Figure 3.2 showcases the mean and standard deviation of length and duration ratios between the source and target sentences and audios. The analyzing the means, it is easily observed that there are little to no differences between on and off-screen. However, the standard deviation of on and off-screen sections is measurably larger. For the scenes where the character is on screen the standard deviation of differences is notably lower than in the off-screen scenes. To make use of that the authors tried to build classifiers that can identify the on and off-screen scenes based on textual features, although the classifiers acted near to random meaning there are no distinct features for this dataset to identify and make a classification for on and off scene annotation.

[1]https://ict.fbk.eu/heroes-on-off/

NMT for Analysis of On-Off Dubbing

As the classification of on and off features did not work, the authors try to explore further distinctions by using NMT models. They use a transformer which is trained with a combination of OpenSubtitles [42], Europarl [11], GlobalVoices [49], MuST-C [19] and WIT3 [12] datasets from the OPUS project [65] and then further fine-tuned by a section of the Heroes [51] dataset. Other than fine-tuning they use a syllabic rescoring method from [62].

The fine-tuned model performed better for all scenarios. It increased the BLEU score for on and mixed sets by around 20% and for off by around 25%. Applying syllabic rescoring on top caused a drop in score with 3 points for on and mixed and 7 points for the off set. It is observable that the off-translation set is less dependent on the dubbing constraints and more on translation quality whilst having a higher increase with fine-tuning. Also shows that adding constraints harms the translation quality of the "off" set the most among others.

3.4. Machine Translation for Dubbing

The author of [10] proposes a method called syllable-rescoring very similar to the phonetic synchrony score in [62] for machine translation systems that are used for dubbing. This method focuses on the isochrony aspect of dubbing. Isochrony in dubbing is the synchronization of the pauses of both source and translated speech parts. It can be simplified down to the match the duration of the original and the dubbed soundtrack. When examined from a textual standpoint, the length of a speech can be approximated by counting the syllables within the corresponding text. Employing syllable-rescoring to increase the chances of a hypothesis sentence being selected if its syllable count is closer to the of the source sentence would amplify isochrony in the dubbing process, aligning speech durations more closely and thereby contributing to a more synchronized and harmonious audio-visual output.

Model & Method

For the architecture, the author chooses the base transformer model from Fairseq [52]. To train the base model the OpenSubtitles 2018 [42] dataset which is a dataset from the Opus project [65] is used. The dataset is partitioned into training and validation subsets, with a distribution ratio of 95% for training and 5% for validation. For testing and fine-tuning, the author opts for the Heroes [51] dubbing dataset. However, to account for the visual dimension inherent in dubbing scenarios, a version is selected in which the dataset is split to on and off screen [36].

Firstly the base transformer model from Fairseq is trained with the OpenSubtitles dataset. Then the MT model is fine-tuned on the Heroes dataset. Then the syllable-rescoring is employed. The model generates N number of hypotheses and within these hypotheses, the rescoring is done as follows:

$$rate(h) = \frac{1}{1 + abs(syll(h) - syll(s))} \tag{3.5}$$

$$newscore(h) = (1 - \alpha) \times score(h) + \alpha \times rate(h) \tag{3.6}$$

In the first function, $syll(x)$ is a function that returns the number of syllables of sentence x, h is the hypothesis sentence and s is the source sentence. In the second function, $score(x)$ denotes the probability of hypothesis x being the translation of the source sentence s, generated by the previously defined MT model.

For evaluation mainly two α's are selected, $\alpha = \{0.3, 0.8\}$. For evaluation metrics, BLEU score [47] and Synchrony score from [62] are used. The fine-tuned model reaches better BLEU scores than the base model however the rescoring has no positive effect on the scores. For synchrony scores the base and fine-tuned models perform much better than the original score of the heroes test set.

4. Integrating Audio Features into Machine Translation

In this chapter, the proposed translation for the dubbing method will be explained. One of the main challenges of dubbing is to make the viewer feel like the character is speaking the dubbed language. This is achieved primarily by matching the translation and the lip movements of the character. To achieve that either the lip movements of the character or the speech should be examined. The main purpose of this study is to understand the speech by automatically learning the audio features and then integrating them into a translation process. This way the end translation will be under the constraints of the given speech file. There can be several methods that achieve this like an End-to-End translation model that is trained with text and audio files that can generate dubbing translations of the target language. However, such a model will require an immense amount of dubbing data.

A review of the works on dubbing in MT in Sections 3.2, 3.3 and 3.4 reveals a common trend among these studies. Specifically, all the examined studies have integrated dubbing constraints by implementing a rescoring mechanism on the MT model's output. This was done by calculating a score between the source and the translation candidate calculating the difference in the number of syllables and then having a weighted average with the generation probability of the candidate. This way the selection process can be easily tuned, either favoring the translation quality or dubbing constraints.

A comparable strategy can be used to integrate the audio features. The concept involves listening to the speech and identifying the translation that sounds the most similar. This can be done by establishing a scoring metric that assesses the similarity of a translation candidate of the target language to a given speech file of the source language. Then the impact of this score can be observed and tuned to achieve an optimal output.

ASR models are capable of extracting and understanding audio features. An ASR model at its core first extracts features of an audio file and then tries to match these features to its previously learned labels. Through forced alignment, predetermined labels can be selected in advance and the probabilities of these labels for an audio file can be extracted.

Given an audio file and translation candidates generated for the transcription of that file, the translation which matches the audio features most, can be prioritized. First, an MT model generates an n-best list of translation candidates given a source sentence. Then for each translation candidate, an alignment score with the source audio file is calculated. This score is calculated by doing a forced alignment. Then using this score a rescoring can be conducted. A visualization of this method can be seen in Figure 4.1.

This adaptive methodology allows for adjustment of the translation process to align with specific objectives related to both translation quality and dubbing constraints. Such a method is presented in the following Methodology Section 4.1.

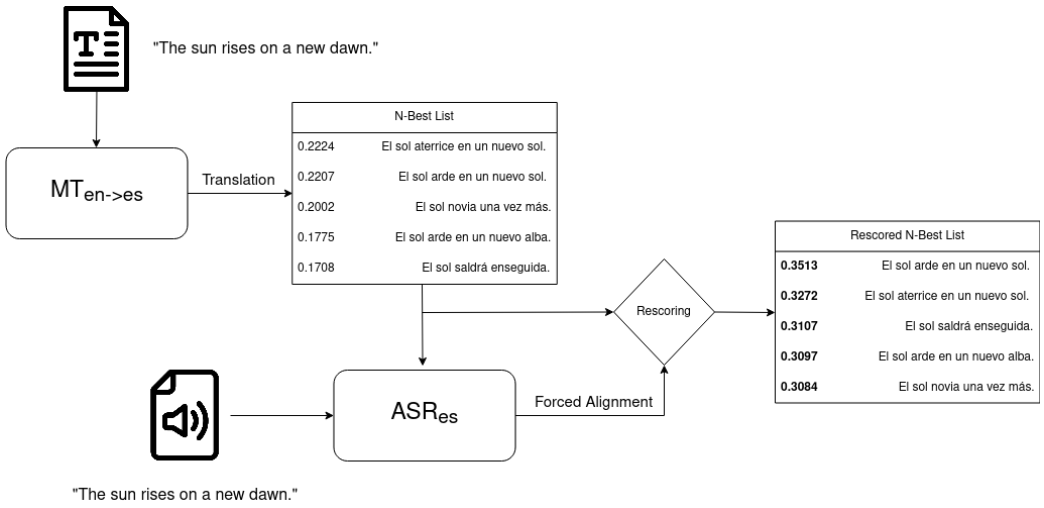


Figure 4.1.: Visualization of the rescoring method.

4.1. Methodology

This study proposes a new asr-rescoring method that prioritizes translations that align most closely with the audio features, thereby contributing to alignment and coherence between the spoken content and its textual representation.

An MT model generates an output translation T_t with probability $P_{T_t}^{MT_{s\to t}}$ of being the translation of a given source sentence T_s:

$$MT_{s\to t}(T_s) = T_t \quad \text{with} \quad P_{T_t}^{MT_{s\to t}} \tag{4.1}$$

where $MT_{s\to t}()$ is the MT model that translates from source language s to target language t. An ASR model generates an output T_s with probability $P_{T_s}^{ASR_s}$ of being the transcript of an audio A_s:

$$ASR_s(A_s) = T_s \quad \text{with} \quad P_{T_s}^{ASR_s} \tag{4.2}$$

where $ASR_s()$ is the ASR function that transcribes audio from source language s. Let's assume a forced alignment function:

$$FA^{ASR}(A, T) = P_T^{ASR} \tag{4.3}$$

where $FA^{ASR}()$ is the forced alignment function for the ASR model. This function receives an audio file A and a transcript T and returns the probability P_T^{ASR} of the transcript T being generated from the given audio A. Since the goal of the method is to find the most fitting translation for dubbing. Translation candidates should be matched with the original audio which is defined as the source audio. Therefore the forced alignment should be made with a target language ASR model ASR_t and between a source language audio A_s and a translated transcript of target language T_t:

$$FA^{ASR_t}(A_s, T_t) = P_{T_t}^{ASR_t} \tag{4.4}$$

where FA^{ASR_t} is the forced alignment function of the ASR model of the language t and $P_{T_t}^{ASR_t}$ is the probability of the transcript being generated by the ASR model ASR_t with the source audio A_s. By doing a weighted averaging of this probability and the probability of the MT model, the rescoring function can then be defined as:

$$P_{rescored} = \alpha \times P_{T_t}^{ASR_t} + (1 - \alpha) \times P_{T_t}^{MT_{s\to t}} \tag{4.5}$$

where α is a hyperparameter. Given a dubbing environment, the only available resources are the source audio A_s and possibly a source transcript T_s. Assuming the MT model generates n best list of translation candidates, the method is then defined as:

$$\begin{aligned} P_{selected} &= max(P_{rescored,i}) && (4.6)\\ &= max(\alpha \times P_{T_{t,i}}^{ASR_t} + (1 - \alpha) \times P_{T_t,i}^{MT_{s\to t}}) && (4.7)\\ &= max(\alpha \times FA^{ASR_t}(A_s, T_{t,i}) + (1 - \alpha) \times P_{T_{t,i}}^{MT_{s\to t}}) && (4.8) \end{aligned}$$

where $T_{t,i}$ are translation candidates generated by the MT model function $MT_{s\to t}()$ from the source transcript T_s with probability $P_{T_{t,i}}^{MT_{s\to t}}$ for $i = 1, 2, ..., n$.

4.2. Forced Alignment

The concept of forced alignment was mentioned in Section 2.3.3. However, not all the ASR models have a built-in forced alignment function. Since the forced alignment is only required for the target language side of the ASR, it is only required with the Spanish language. As Spanish ASR models, Whisper-large and Wav2Vec2-voxpopuli models are used.

4.2.1. Wav2Vec2

With the use of the library WhisperX [9], it is feasible to emulate a forced alignment with the Wav2Vec2 ASR model. As mentioned in Section 3.1, WhisperX utilizes two ASR models, one for transcription and the other one for the alignment task. The transcription model is a Whisper model and the alignment model is a Wav2Vec2 model. The speech file is first transcribed and segmented by the first source ASR model which is the Whisper-largev2 model. The difference with Whisper-large is that this new model is trained for 2.5 more epochs with data augmentation. The output result of the model includes multiple segments from the audio file, however, the Heroes dataset includes pre-segmented speech files by sentences, so multiple segments of an audio file are not considered during this study. Each segment's transcription, the start and end of the utterance are noted in this result. Also,

the language that is transcribed is included, however, we do not need that information currently. An example output result can be seen in Listing 4.1. In this example the text is given as "The sun rises on a new dawn.", the start and end of the utterance are given as 0.008 and 2.101. So the utterance of the sentence "The sun rises on a new dawn." started at 0.008s and ended at 2.101s of that speech file.

```
{'segments':
    [
        {'text': ' The sun rises on a new dawn.',
        'start': 0.008,
        'end': 2.101
        }
    ],
    'language': 'en'
}
```

Listing 4.1: The output result of the first ASR model.

This result is passed to the alignment model and a word-level segmentation with its probabilities is calculated. As with the first output results it includes the complete sentence and the start and end time of the utterance. Also outputs the word segments that go through the sentence word by word and for each word the start, end, and probability of the utterance between the given time intervals being the word.

The model aligns each word of the sentence, to time intervals in from the speech file and also gives out probabilities. An example alignment can be seen in Listing 4.2. Here the sentence "The sun rises on a new dawn." is fragmented into words and then the words are aligned with the time frames of the speech file. In the word_segments part of the output each word and their matched utterance time frames can be observed. For each word and time frame the model calculates a probability of that word being that utterance. The word with the highest probability is "dawn." with a 1.0 following that "new" is also well understood being 0.936 probability. The most uncertain word is "a" with a 0.586 probability. Looking at the example it can be said that the model matches the words with a very high confidence.

By accessing and changing the 'text' from the results in Listing 4.1, the alignment model can be forced to make the alignment of the new sentence with the given audio. The use of a target language (Spanish) ASR model (Wav2Vec2-voxpopuli) for the alignment ASR model, allows the text to be given in the target language. The Listing 4.3 showcases an example result output from the alignment as model. As input the text "The sun rises on a new dawn." is changed with the corresponding translation "El sol ilumina un nuevo amanecer." from the first models' output. Then the modified output segment is given as input to the alignment model. The alignment model dissects the sentence into words and aligns each word with a time frame. For each time frame, it calculates the probabilities of each utterance being the given word for that frame. However, a probability of that sentence being the transcript is not provided.

```
{'segments':
    [
        {'start': 0.048,
        'end': 1.819,
        'text': ' The sun rises on a new dawn.',
        .
        .
        .
    ],
'word_segments':
    [
        {'word': 'The', 'start': 0.048, 'end': 0.149, 'score': 0.851},
        {'word': 'sun', 'start': 0.209, 'end': 0.491, 'score': 0.916},
        {'word': 'rises', 'start': 0.592, 'end': 1.034, 'score': 0.745},
        {'word': 'on', 'start': 1.135, 'end': 1.195, 'score': 0.93},
        {'word': 'a', 'start': 1.256, 'end': 1.296, 'score': 0.586},
        {'word': 'new', 'start': 1.336, 'end': 1.517, 'score': 0.936},
        {'word': 'dawn.', 'start': 1.558, 'end': 1.819, 'score': 1.0}
    ]
}
```

Listing 4.2: The output result of the alignment ASR model.

```
{'segments':
    [
        {'start': 0.048,
        'end': 1.638,
        'text': 'El sol ilumina un nuevo amanecer.',
        .
        .
        .
    ],
'word_segments':
    [
        {'word': 'El', 'start': 0.048, 'end': 0.249, 'score': 0.672},
        {'word': 'sol', 'start': 0.29, 'end': 0.471, 'score': 0.386},
        {'word': 'ilumina', 'start': 0.491, 'end': 0.873, 'score': 0.318},
        {'word': 'un', 'start': 0.914, 'end': 1.034, 'score': 0.436},
        {'word': 'nuevo', 'start': 1.075, 'end': 1.296, 'score': 0.379},
        {'word': 'amanecer.', 'start': 1.316, 'end': 1.638, 'score': 0.279}
    ]
}
```

Listing 4.3: The output result of the alignment ASR model.

In the Example 4.3, the word segments are given. The sentence "El sol ilumina un nuevo amanecer.", was dissected to the words "El", "sol", "ilumina", "un", "nuevo", and "amancer.". Between each word, the word with the highest probability is "El" with 0.672 and the lowest is the "amanecer." with 0.279. Comparing the probability of this to results in the Listing 4.2, the decline in probabilities is observable and it is expected. Naturally listening to an English speech and trying to understand the Spanish words from it will not yield as high of a confidence as the English ASR. However, it is an important observation that the model gives different time segments for the words in each example. It means that the model is not initially framing the utterances and trying to match the words to these utterances. Rather the model checks the words and tries to find an utterance that can most fit the word. It is an interesting point and looking at both Listings 4.2, 4.3, the word "El" and "The" is the first word of both text inputs but the time segments that the model selects for both words are not the same. The word "El" starts and ends between 0.048-0.249 and the word "The" starts and ends between 0.048-0.149. There is a very significant 0.1 difference between the ending of both words. This can push the start and end of the next words and cause the upcoming words to have a worse alignment. For example the phonological similarity between the word "sol" and the word "sun" than the similarity of words "El" and "The". However, the probability of the word "sol" is 0.386 and is significantly lower than the word "El".

Even though the alignment of the text on a foreign language audio might not be perfect, it still provides significant importance. The probabilities are given for words and not for the complete sentence. Therefore, the average probability of the words is taken as an estimate to represent the probability of the sentence. This is namely an estimate because each word might not have the same importance for the integrity and meaning of the sentence. This taken estimate probability is then used as the forced alignment score mentioned in Section 4.1.

4.2.2. Whisper

WhisperX offers various pre-trained Wav2Vec2 models for the forced alignment task. Whisper is a fairly new model, and it was released in December 2022. Even though it got popular very quickly, there are no tools that allow a forced alignment. Therefore for this study, a forced alignment module for Whisper is implemented.

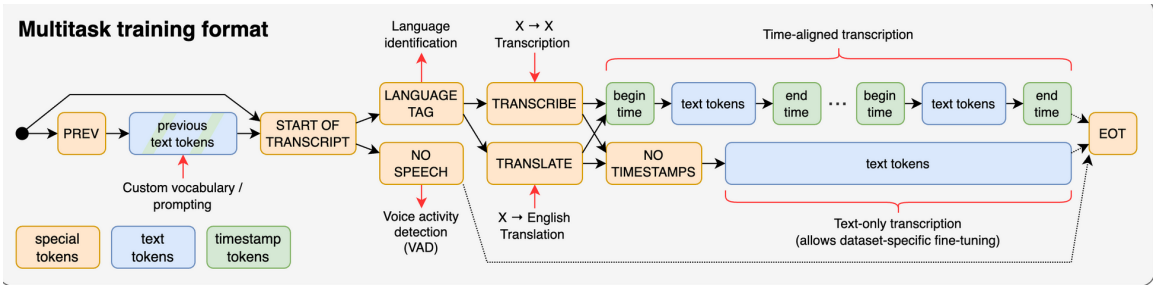


Figure 4.2.: Whisper's pipeline for decoding and transcriptions [58].

```
DecodingResult(
    audio_features=tensor([[...]], device='cuda:0', dtype=torch.float16),
    language='es', language_probs=None,
    tokens=[50364, 2699, 1404, 1930, 449, 1426, 517, 18591, 669, 1929, 1776,
            13, 50464],
    text='El sol ilumina un nuevo amanecer.',
    avg_logprob=-1.5413819721766882,
    no_speech_prob=0.04618218168616295,
    temperature=0.0,
    compression_ratio=0.8048780487804879)
```

Listing 4.4: The output result of the Whisper used for forced alignment.

Firstly, the workflow of Whisper should be investigated. In Section 2.3.2.4, the multitask format of Whisper and its pipeline is explained. The Figure 4.2 showcases the pipeline of Whisper. Whisper starts by detecting the language of the given speech file. Since in this study, for the forced alignment task, the text is going to be in a different language the audio file, should be set through the parameters rather than letting the system do it automatically. Then the system starts going through audios and starts predicting timestamps. The timestamps are predicted relative to the current audio segment. They are quantized to the nearest 20 milliseconds. For these quantized time segments, the Whisper model predicts the matching token. This token prediction is done in the decoder. Whisper currently has 2 decoding modes, a greedy decoder, and a beam search decoder. To do forced alignment a new alignment decoder is implemented into Whisper's decoding.py file [1]. A detailed implementation guide can be found in Section A.1.

The forced process requires a text and an audio file as input. Whisper first quantizes the audio file into segments and then loops the segments within the decoders. The alignment decoder converts each word in the input string into tokens. Then for each audio segment, it checks if the audio segment matches a blank token, if not it forces a predefined word token and calculates the probabilities. The process continues until there are no more words left or all the audio frames are looped. In the end, the decoder inserts a `<|endoftranscript|>` token and concludes the process.

The alignment decoder provides Whisper with a forced alignment function. A forced alignment module was already available for Wav2Vec2 model. However, Whisper is a larger model than the Wav2Vec2 model and therefore an evaluation of various sized models would provide a more robust result to asses the effect of this methodology. This functionality can calculate the probabilities of the text tokens with matching audio frames, providing a similarity measure between the audio and the text. Whisper uses a Byte-Pair-Encoding (BPE) level tokenizer, the probabilities of each BPE token is calculated and aggregated. For the case of this study word-level probabilities are not required since the purpose is to score each translated sentence based on its similarity to a given speech file.

[1] https://github.com/openai/whisper/blob/main/whisper/decoding.py

A Brief Comparison

The Listing 4.4 showcases an example result of the Whisper model when run with the alignment decoder. The input for this example run is the same as in the Listing 4.3. With the Whisper ASR model for aligning the probability equates to around 0.2140 and with the Wav2Vec2 ASR model the word probabilities average around 0.4116. This is a major difference between both models. Further evaluation and comparison of both models' effect on dubbing will be done in Chapter 5.

4.2.3. Integration of the Alignment Scores

It is observed that after rescoring the newly selected translations always stayed the same for all given α values. Upon further investigation, it was discovered that the scores obtained from forced alignment are either higher than the mt-probabilities or very low completely not affecting the outcome. The discrepancy between the scores for MT and ASR systems were not similar. This inherently makes sense because of MT and ASR models implicitly have different scoring techniques. This can be further observed in the following examples.

Mt-score	Asr-score	Rescored	Sentence
0.569113	**0.545400**	**0.563184**	**La gente no, Claire, lagartos.**
0.563033	0.526	0.553775	No la gente, Claire, lagartos.
0.578234	0.46175	0.549113	No personas, Claire, lagartos.
0.573807	0.475	0.549106	No gente, Claire, lagartos.
0.496777	0.518833	0.502291	La gente no, Claire, los lagartos.

Table 4.1.: An example where the Asr-score is relevant and impacts the selection.

The Table 4.1 shows an example where the rescoring had an impact on the outcome. However, this example is very rarely seen throughout the dataset, having both mt- and asr-scores very closely separated. However, a more common occurrence can be observed in Table 4.2. Even though the highest asr-score is nearly 2.5 times the second highest score it can have no effect on the rescoring for any feasible α there is.

Mt-score	Asr-score	Rescored	Sentence
0.501758	0.06825	0.285004	¿Qué te pasa, amigo?
0.377794	**0.09125**	0.234522	- ¿Qué pasa, amigo?
0.622514	0.039667	**0.331091**	¿Qué pasa, amigo?
0.599922	0.043	0.321461	¿Qué sucede, amigo?
0.503048	0.036	0.269524	¿Qué ocurre, amigo?

Table 4.2.: An example where the Asr-score is not significant for the rescoring.

In order to make the discrepancy between alignment scores more distinct, the alignment scores are normalized. Only the alignment scores are normalized, so that the discrepancy between the scores are highlighted and the method can decide between the candidates focusing more on the vocal similarity rather than the translation quality which the MT

scores represents. Two different normalization techniques are used. One being a local and the other one being a global normalization. The general normalization calculation is defined as:

$$NormalizedScore = \frac{asrScore - min(scores)}{max(scores) - min(scores)} \tag{4.9}$$

where *asrScore* is the score defined in Equation 4.4 and the *scores* are defined in context of local or global. The local min and max scores are calculated within translation candidates of the same source sentence. The global min and max are calculated within scores from the whole test dataset. The differences between locally and globally normalized scoring is further analyzed and evaluated in Chapter 5.

4.3. Experimental Setup

4.3.1. Datasets

To have a strong MT model tailored for dubbing, the model should be contextually adapted to movies and series. Opus [65] project is a valuable resource, providing a dataset that suits this purpose. The dataset OpenSubtitles-2016 [42] includes parallel text data, where each subtitle corresponds to a specific segment of a movie or TV show and is aligned with translations in multiple languages. This study will make use of the English-Spanish parallel data from the corpus. The statistics of the en-es parallel dataset can be observed from Table 4.3.

OpenSubtitles en-es	
documents	62,2k
sentences	61.434.252
words	756,26M

Table 4.3.: The statistics of OpenSubtitles en-es parallel corpus.

For fine-tuning and testing Heroes corpus [51] is used. Heroes corpus is generated from a 2006-2010 sci-fi drama TV series called "Heroes". The series was originally filmed in English and dubbed into multiple languages, one of which is Spanish. The dataset contains parallely aligned English-to-Spanish single-speaker speech segments with their transcriptions. It is extracted from 21 episodes of seasons 2 and 3 of the TV show. The dataset contains nearly 5 hours of parallel data in which each speech segment averages 2.44 seconds in English and 2.42 seconds in Spanish. The statistics of the Heroes dataset can be observed from the Tables 4.4.

4.3.1.1. Scenarios

With the current data and resources in hand, a dubbing scenario should be imagined. Previous related works [36, 62, 10] have handled the experiments only with the use of the text data from the Heroes dataset. They never made use of any ASR models or systems that

	English	Spanish
sentences	10241	9771
words	57768	50006
tokens	70744	64905

	English	Spanish
avg. word/sentence	5.64	5.11
avg. word/segment	7.99	6.92
avg. sentence/segment	1.41	1.35

Table 4.4.: Statistics of Heroes dataset.

utilized the speech part of the dataset. In this study, as previously mentioned in Section 4.1 a methodology is proposed which uses a forced alignment method that utilizes the speech data and the text data together. This can be done when the audio and the text of both source language is available. However, that might not always be the case.

In some cases, the source transcript might not be available. In these scenarios, the transcript should be extracted from the speech file. Since the source language in the Heroes dataset is English, an English-capable ASR should be employed. The selection of such an ASR model is discussed in the following Section 4.3.5.

4.3.2. Tools and Frameworks

This study is done with Fairseq [52]. Fairseq is a Python-based sequence modeling toolkit containing libraries for preprocessing and evaluation as well as model architectures and pre-trained models designed for NLP tasks. Fairseq's modeling tasks are handled by Pytorch [55]. For faster and more efficient training and generation phases, Nvidia's CUDA [17] library utilizes GPUs.

To reduce training time the pretrained model DeltaLM [45] is used. DeltaLM has a similar architecture to the vanilla transformer from [66]. However, its decoder is extended by an FFN layer connecting the self-attention layer to the cross-attention layer. DeltaLM is capable of doing various NLP tasks such as text summarization, question answering, and machine translation. The statistics of DeltaLM can be further investigated in Table 4.5.

DeltaLM	
encoder layers	12
decoder layers	6
attention heads	12
FFN layers	3072
total parameters	360M

Table 4.5.: Layer statistics and parameter count of DeltaLM pre-trained model.

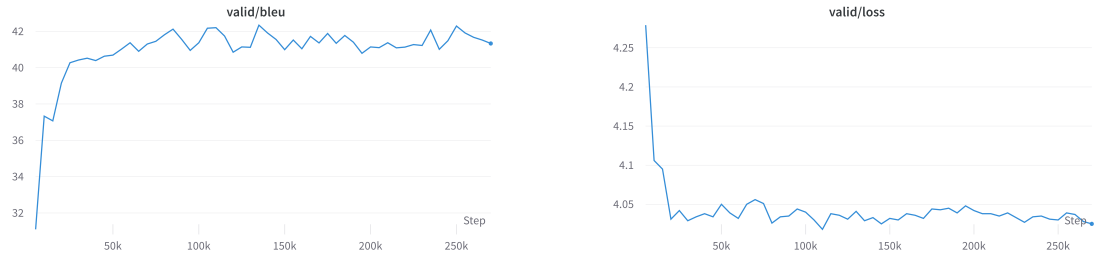
An implementation of DeltaLM[2] with Fairseq facilitates the base of this study to train and generate translations. To integrate the audio features ASR models Wav2Vec2, Whisper, and WhisperX mentioned in sections 2.3.1, 2.3.2 and 3.1 are employed.

[2]https://github.com/microsoft/unilm/tree/master/deltalm

4.3.3. Data Preprocessing & Training

Firstly a baseline model is necessary to make a comparison with the rescored method. The baseline model must be competitive as its own, so that it can provide high-quality translation candidates the method can then rescore and choose from. Therefore a well pretrained model is used. DeltaLM is a pre-trained model for various natural language processing tasks however, for this study the main focus will be doing translations for dubbing, adapting the context of the MT model can yield better translation quality. This is achieved by fine-tuning the model with data from the OpenSubtitles dataset. The dataset is first portioned into 95% training and 5% validation sets. The datasets are then tokenized using Sentencepiece[39]. The Sentencepiece model[3] is provided by DeltaLM. The tokenized data is then binarized using the preprocess method of Fairseq. The dictionary used in preprocessing is also provided by DeltaLM.

The next step is to fine-tune the model on the preprocessed data. The model is fine-tuned with Adam optimizer with $\beta = (0.9, 0.98)$, the learning rate starts with 0.0000001 and increases to 0.0001 in 4000 warm-up steps. Then the learning rate decays with inverse square root through the training. Label smoothed cross-entropy is used with 0.1 label smoothing. The model is then trained for 270k steps.



(a) BLEU scores of validation set during training.

(b) Loss values of validation set during training.

Figure 4.3.: Fine-tuning with OpenSubtitles for 270k steps.

The model reaches its best BLEU score on the validation set around 140k steps. Before fine-tuning the model had a 37.33 BLEU score on the validation set and it is increased to 42.34. An improvement of around 13%. The loss value peaked at around 20k steps and stayed similar. The graphs showing the progression of the fine-tuning can be observed in Figures 4.3

Additionally, the model is fine-tuned on the Heroes dataset. Heroes dataset is split into "on", "off" and "mixed" segments as mentioned in Section 3.3. The segments are further separated into test, validation, and training parts. First 400 randomly selected unique sentences are selected for the test sets. 10% of the remaining sentences are sampled randomly for the validation sets, and the rest are the training sets. The randomly sampled indexes are saved into a file to keep the random seed the same for the integrity of all experiments. To match both scenarios mentioned in Section 4.3.1.1 these indexes are used for matching audio files so that transcribing the audio subsequently will yield the same

[3]https://deltalm.blob.core.windows.net/deltalm/spm.model

test set used in the first scenario with all sentences being transcriptions. It is expected that the transcription test set would be of lower quality, and therefore the translations generated for these sentences will also lack quality.

To use for training and experimentation an aggregate set "all" a sum of "on", "off" and "mixed" is created through the concatenation of the subsets. The model is fine-tuned with the training set from the "all" segment. The same fine-tuning method is applied. The model is fine-tuned for an additional 35k steps.

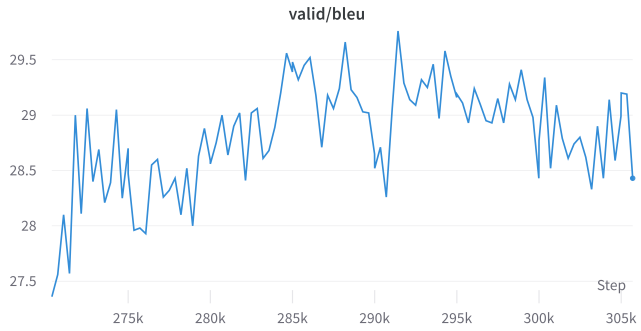


Figure 4.4.: The progress of the model performance on the validation set in the fine-tuning with Heroes dataset for 35k steps.

The model starts performing around 27.3 BLEU score on the validation set at the start of the fine-tuning. It reaches its maximum of 29.7 and further oscillates down to 28.5 at the end. The training set of Heroes is 5200 sentences and the gain in performance is therefore relatively small. The main purpose of this fine-tuning works more towards shaping the domain of the model into dubbing.

4.3.4. MT Model Comparison

The purpose of the experiment is to integrate the audio features into the MT process to enhance the dubbing quality. Previously integration of dubbing constraints was achieved by reranking the output of the MT model by some metrics see Sections 3.3, 3.4. Therefore having a good-performing MT model is crucial, which can output various high-quality translations that can undergo rescoring based on specified criteria.

	on	off	mixed	all
Transformer-FT_{heroes}[10]	16.7	19	15.7	16.8
Transformer-FT_{heroes}[36]	26.3	30.84	26.33	27.5
DeltaLM-$FT_{opensub}$	24.94	25.37	20.69	23.13
DeltaLM-$FT_{+heroes}$	**29.85**	**32.65**	**28.84**	**30.17**

Table 4.6.: BLEU score comparison of fine-tuned models on the heroes test dataset.

Table 4.6 shows that the DeltaLM-$FT_{+heroes}$ model has the best BLEU scores considering the heroes test dataset. This does not guarantee that the model is best performing since

each paper has a randomly sampled test set and the seeds are not provided. However, it is safe to assume that the double-fine-tuned DeltaLM model is very competitive and capable of generating high-quality translations. The `DeltaLM-`$FT_{+heroes}$ performs better than its previous version and is used as the baseline model in the experiments.

4.3.5. ASR Model Selection

In the context of a dubbing environment, the presence of audio content is assured, but the availability of a human-generated transcript may not always be guaranteed. To simulate such an environment a scenario is realized where the only accessible resource is the audio. To create a transcript for the MT model, an ASR model of the source language is needed. Additionally, to do forced alignment an ASR model of the target language is required. Therefore an analysis of the ASR models available is crucial since the quality of the transcription can have an impact on the outcoming results.

	Whisper					wav2vec2	
	tiny	base	small	medium	large	largelv60k960h	voxpopulibase
Eng	0.3629	0.2833	0.2175	0.1886	0.1755	0.4263	-
Eng case&punc	0.2779	0.1985	0.1420	0.1153	0.1067	0.2878	-
Spa	0.6048	0.4263	0.2833	0.2266	0.2167	-	0.6227
Spa case&punc	0.5104	0.3657	0.1935	0.1375	0.1303	-	0.5516

Table 4.7.: The word error rate (WER) comparison of different ASR models. All of the audio and corresponding transcriptions of Heroes corpus are used as test data.

Table 4.7 showcases an analysis of various ASR models on the Heroes corpus. Tiny, base, small, medium, and large are Whisper models, and two pre-trained Wav2Vec2 models are used. For English WAV2VEC2_ASR_LARGE_LV60K_960H and Spanish VOXPOPULI_ASR_BASE_10K_ES pretrained models from Pytorch[4] are used. The references are punctuated and since Whisper has a built-in case and punctuation tool and Wav2Vec2 models do not, two different comparisons are done. One where the outputs are directly compared with the references and one where the outputs and references are first lowercased and then the punctuations are removed from each sentence by a Python script. This way a more fair word error rate (WER) comparison can be done where the punctuation does not create additional tokens extending the words.

Of all the models Whisper-large model is the highest performer. It is the biggest model within Whisper's selection and therefore the slowest. For the source (English) language ASR model Whisper-large is selected, it creates the highest quality transcriptions for cased&punctuated which is imperative to have as references for the MT model. For the target (Spanish) language ASR model both Whisper-large and voxpopulibase are employed. While the Whisper-large model brings a high-quality understanding of the audio features, the voxpopulibase is very lightweight and even though the decoder might not produce great transcriptions, the encoder could be well-trained from the pre-training.

[4]https://pytorch.org/audio/stable/pipelines.html

5. Evaluation

In this chapter, an evaluation of two methods will be done. The first method is the syllable-rescoring which was mentioned in Sections 3.3, and 3.4. The second method is the asr-rescoring which was proposed by this study and detailedly mentioned in Chapter 4. For evaluation, the metrics BLEU, COMET mentioned in Section 2.2.4 and the synchrony-score mentioned in Section 3.2 are used. From Section 2.2.4, it is known that the COMET metric performs closer to a human evaluation than the BLEU. While BLEU tries to find direct word-per-word matches between the reference and hypothesis sentences, COMET employs an MT system that considers the source, reference, and hypothesis. Therefore each metric has a different aim and meaning for this evaluation:

- BLEU: This metric can be viewed as the dubbing quality, as it measures the similarity between the selected translation and the original dubbed translation. It showcases the overlap between the generated translation and the reference dubbed text.
- COMET: This metric can be viewed as the translation quality, as it assesses how well the output aligns with both the source and reference sentences. It provides insights into the overall translation performance and the ability of the model to generate linguistically accurate content.
- Synchrony-score: This metric can be viewed as a measurement of dubbing constraints. It serves as a measure of how constrained the generated content is, showcasing an estimated phonetical duration ratio for the source and the translation.

These metrics serve as quantifiable measures to assess the effectiveness and performance of the respective methods in achieving their intended objectives.

5.1. Hyperparameter analysis

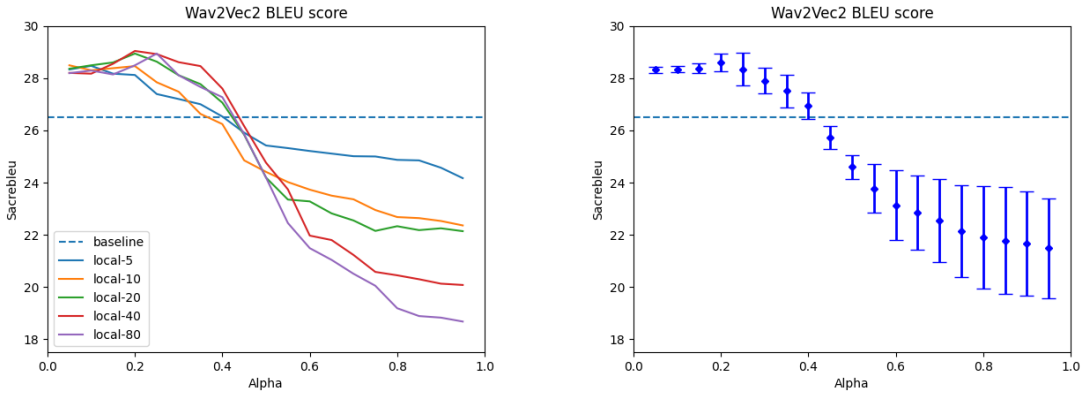
A key factor to the rescoring is the selection of the hyperparameter α. If $\alpha = 1$ is set, the selection only depends on the forced alignment score and if $\alpha = 0$ is selected, the probability of the MT model is selected, making the rescoring pointless. A balanced parameter that positively influences the model's performance should be selected. Therefore an analysis is conducted. A linear searching method is used on the validation set for all $\alpha = 0.05, 0.1, 0.15, ..., 0.95$ to find the best performing α values for each case. For this analysis, a validation set is used instead of the test set to prevent overfitting the α. Further, this analysis is conducted on the validation set of the "on" segment because it is substantial that this rescoring benefits the dubbing where the actively speaking character is on-screen. Then for each α the BLEU scores, COMET scores, and synchrony scores are compared

to find the most fitting α. For the comparison below the translations for $n = 5, 10, 20$ are generated by beam size 20 and for $n = 40, 80$ are generated respectively by beam sizes 40 and 80. The baseline score of $n = 10$ is used, even though there were differences between baseline scores among different beam sizes, the differences were small and insignificant.

5.1.1. Wav2Vec2

Figure 5.1a showcases the effect of the hyperparameter and the n-best list size on the BLEU score. This method uses Wav2Vec2 as the ASR model and uses local normalization to calculate the asr-scores. The cases are separated by n-best list sizes. The x-axis represents the α and the y-axis represents the BLEU score. The dotted blue line represents the BLEU score of the translation that the MT model produces while the other lines are cases with different n-best list sizes. At some point in the graph, each line surpasses the baseline. This happens for $\alpha < 0.4$. All the cases then decline in scores extremely steeply around $\alpha = 0.5$. The highest BLEU score reached between all cases is 29.04 with $\alpha = 0.2$ by $n = 40$. This is an approximately 10% increase over the baseline.

From mean and standard deviation Figure 5.1b, a general view of the distribution and the effect of the α value over the BLEU scores can be seen. For the selection of α, performance and consistency are important. Luckily for this case, the highest BLEU score yielding $\alpha = 0.2$ also is the most consistent high scorer with a mean value of 28.61. A comparison can be made to $\alpha = 0.25$, however, the standard deviation of $\alpha = 0.25$ is higher and therefore can perform lower for the test case.



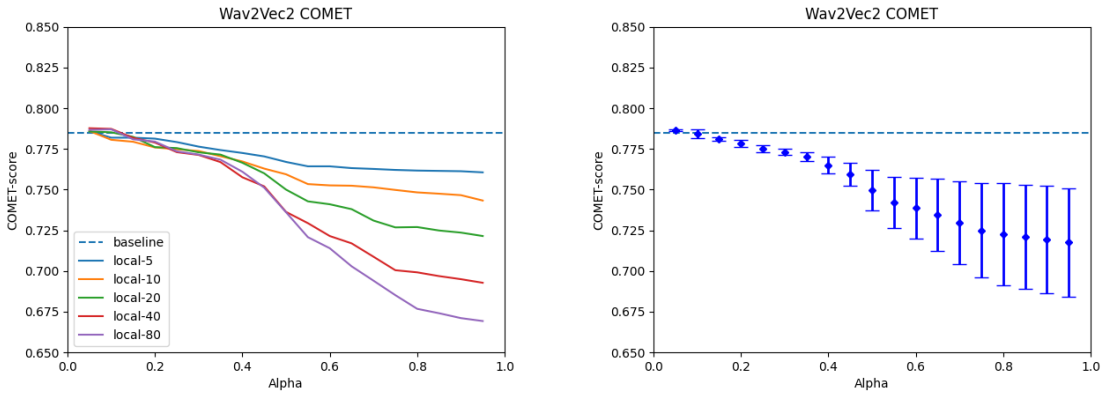
(a) BLEU scores of N-best.

(b) Mean and standard deviation of N-bests.

Figure 5.1.: The distribution of BLEU scores over the α values for rescoring done with locally normalized asr-scores.

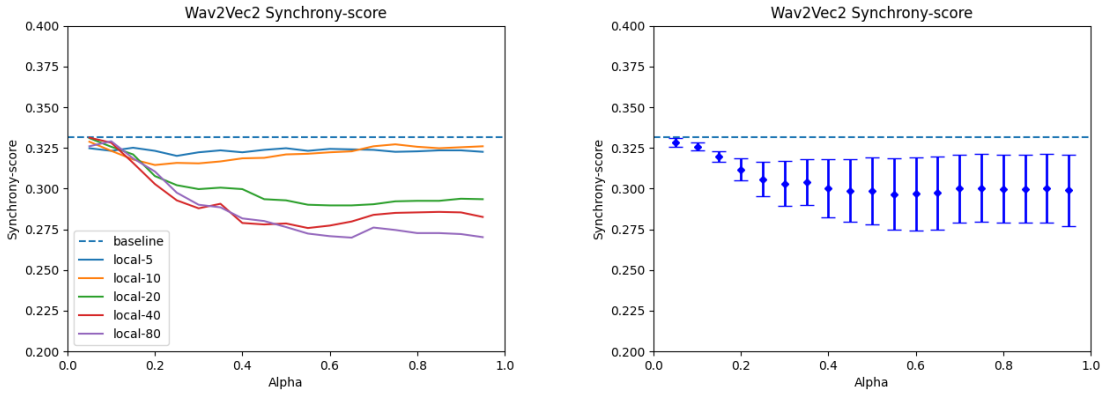
Figure 5.2a represents the COMET score distribution over the α values. It can be observed that only for $\alpha = 0.05$ are the cases performing better than the baseline. Higher n-best list size the scores are affected more by the α selection having a higher oscillation. Interestingly the BLEU score promises increases at a higher percentage and for a wider range of α values, however looking at the COMET scores a similar picture is not seen. The best score is 0.7877 with $\alpha = 0.05$ and $n = 40$.

Looking at the consistency of COMET scores for different α values within various n-best list sizes in Figure 5.2b. It can be observed that as the α value grows increasing the effect of the ASR models scoring, the translation quality and consistency decrease. The most consistent and the average highest score is $\alpha = 0.05$.



(a) COMET scores of N-best. (b) Mean and standard deviation of N-bests.

Figure 5.2.: The distribution of COMET scores over the α values for rescoring done with locally normalized asr-scores.

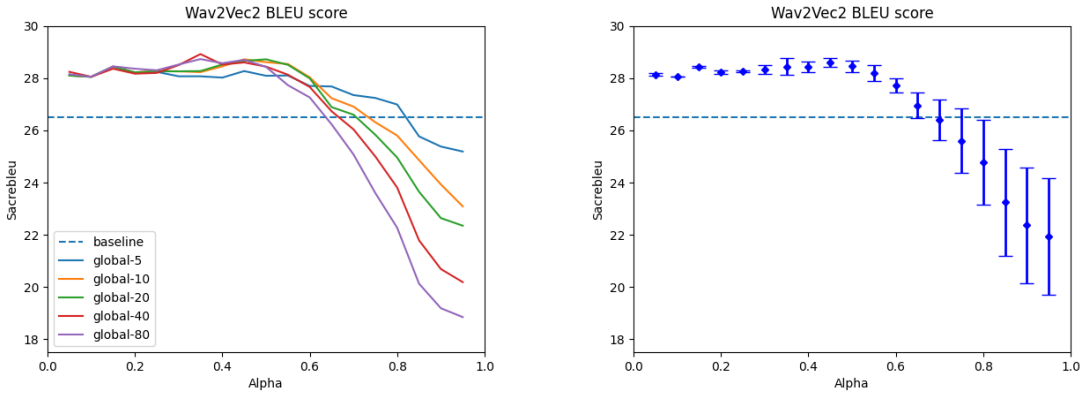


(a) Synchrony-scores of N-best. (b) Mean and standard deviation of N-bests.

Figure 5.3.: The distribution of synchrony-scores over the α values for rescoring done with locally normalized asr-scores.

Figure 5.3a represents the distribution of synchrony scores over the α values for differently n-best list-sized cases for the Wav2Vec2 model. The baseline synchrony score is calculated between the source and the translation provided by the baseline model. All the methods select better translation candidates than the baseline. The worst performing cases are $n = 5, 10$. As the n-best list size increases the synchrony score decreases. The best synchrony-score is 0.2699 for $\alpha = 0.65$ by $n = 80$. This represents approximately a 7% decrease from the baseline synchrony score.

From Figure 5.3b, it can be seen that for all of the α values, this method consistently performs better than the baseline. Interestingly the best scoring α for this case is not the best selection over all the n-best list sizes. The average performance of $\alpha = 0.6$ seems to be slightly better than the $\alpha = 0.65$.



(a) BLEU scores of N-best.

(b) Mean and standard deviation of N-bests.

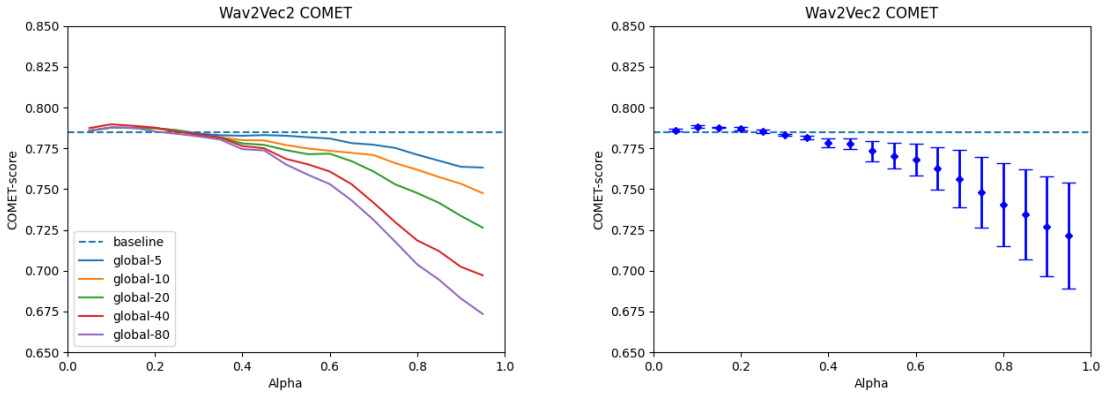
Figure 5.4.: The distribution of BLEU scores over the α values for rescoring done with globally normalized asr-scores.

The Figure 5.4a represents the distribution of BLEU score over α's for differently n-best list-sized cases for a rescoring done with globally normalized scores from the Wav2Vec2 model. The BLEU score distribution over the α values is more consistent than the locally normalized ones. All the cases perform better than the baseline for $\alpha < 0.65$. As the α increases, the performance of the method decreases rapidly. This decrease is higher for larger n-best list sizes. Excluding $n = 5$ all the other sizes perform very similarly at their peaks. The highest score achieved by the cases $n = 10, 20$ is 28.72 respectively with $\alpha = 0.45, 0.5$. For $n = 80$ the highest score is 28.73 with $\alpha = 0.35$ and the overall highest score is 28.92 approximately a 9% increase over the baseline achieved by $n = 40$ with $\alpha = 0.35$.

Figure 5.4b shows the mean and standard deviation distribution of globally normalized rescoring of Wav2Vec2 models. The BLEU scores for $\alpha > 0.7$ are low and inconsistent. Looking at the averages, the highest performing α value is 0.45. Even though the highest BLEU score reached is with $\alpha = 0.35$, $\alpha = 0.45$ yields higher results more consistently with a mean of 28.59, and therefore is selected as the α value for the globally normalized test set results.

Figure 5.5a showcases COMET scores for the globally normalized Wav2Vec2 cases. The globally normalized scores are not higher than the locally normalized ones. However, similarly to the difference in BLEU graphs, the scores are a little more consistent among a wider range of α values. Another similar pattern is the oscillations of higher n-best list sizes are also higher. The highest achieved score is 0.7897 with $\alpha = 0.1$ and $n = 40$.

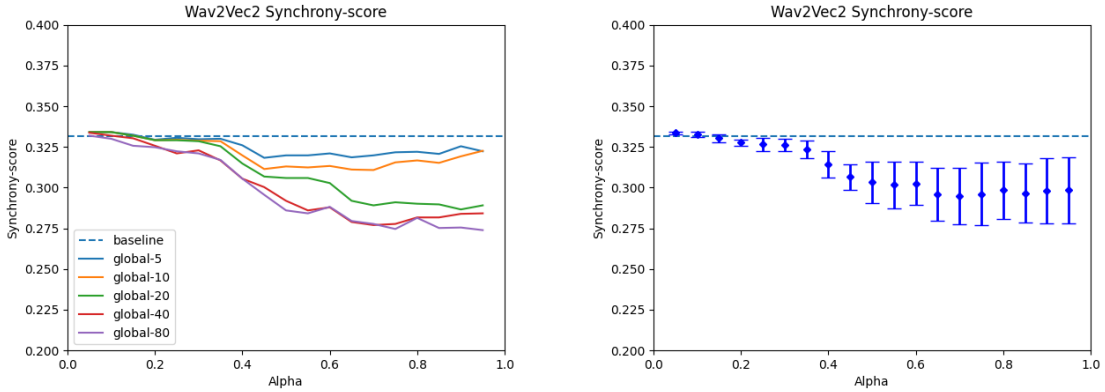
Figure 5.5b represents the mean and standard deviation of COMET scores among the n-best list sizes. It is observable that for higher α values the quality and the consistency of



(a) COMET scores of N-best. (b) Mean and standard deviation of N-bests.

Figure 5.5.: The distribution of COMET scores over the α values for rescoring done with globally normalized asr-scores.

the translation is decreasing similar to the locally normalized case. The highest and most consistent score is at $\alpha = 0.1$.



(a) Synchrony-scores of N-best. (b) Mean and standard deviation of N-bests.

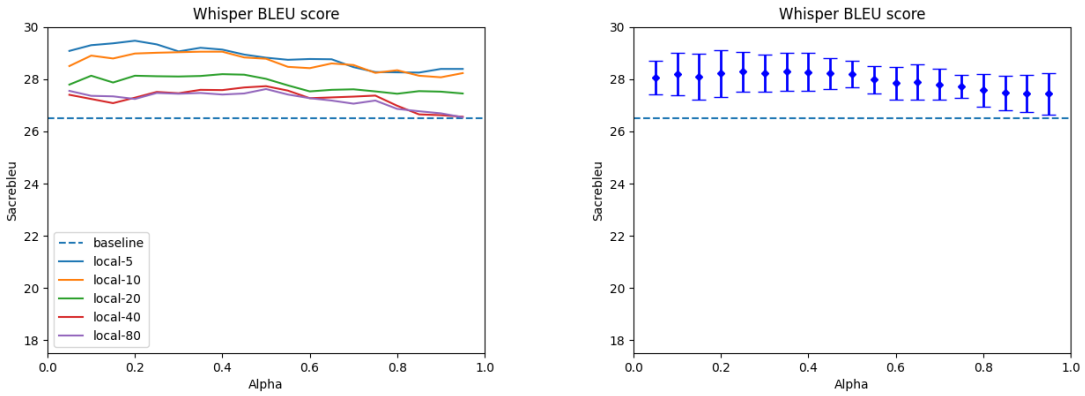
Figure 5.6.: The distribution of synchrony-scores over the α values for rescoring done with globally normalized asr-scores.

From Figure 5.6a, the synchrony scores for the globally normalized rescoring of the Wav2Vec2 model can be observed. Similar to the locally normalized rescoring Figure 5.3a, the synchrony scores are very close to the baseline for $\alpha < 0.45$. The cases with $n = 5, 10$ are the worst-performing cases. The cases $n = 40, 80$ perform the best among all cases. The best synchrony-score 0.2739 approximately a 5% decrease from the baseline is achieved by $n = 80$ for $\alpha = 0.95$.

Lastly, Figure 5.6b showcases the mean synchrony-score distribution of n-best list sizes over the α values. The average of the cases performs better than the baseline for $\alpha > 0.1$. The most consistently performing α value would be 0.7 with an average of 0.29488.

Further looking at all the cases, it can be observed that the alignment scores calculated with the Wav2Vec2 model has a positive effect on the BLEU score for several α values. The globally normalized version seems to be more consistent than the locally normalized version. Observing synchrony scores, it is apparent that the method can learn the synchrony constraint. Additionally examining the COMET scores, it can be seen that the model does not have any positive effect on the translation quality.

5.1.2. Whisper



(a) BLEU scores of N-best.

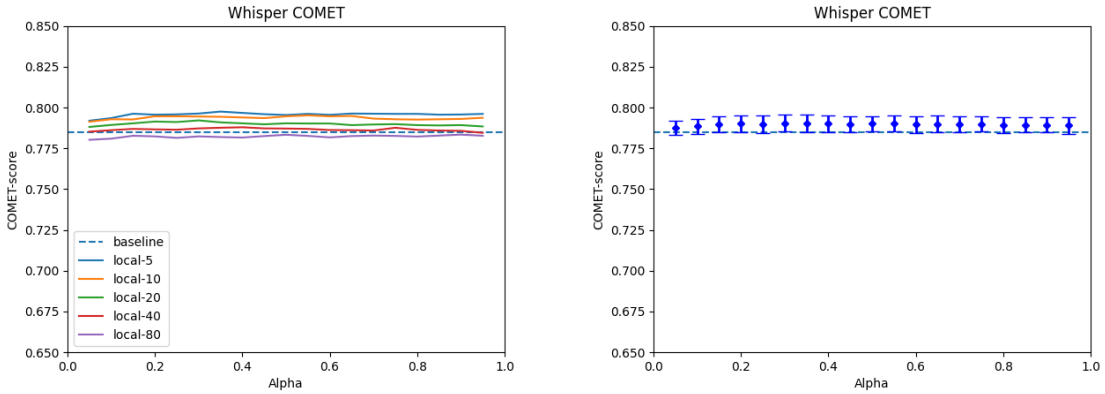
(b) Mean and standard deviation of N-bests.

Figure 5.7.: The distribution of BLEU scores over the α values for rescoring done with locally normalized asr-scores.

Figure 5.7a represents the distribution of BLEU scores over the α values for different n-best list sizes. The outcome of rescoring resulted in a BLEU score increase over the baseline. There is a distinct performance difference between the sizes $n = 5, 10$ and $n = 20, 40, 80$. In contrast to the results of Wav2Vec2, the lower n-best list sizes perform better with Whisper's rescoring. The highest performing case for Wav2Vec2 $n = 40$ is interestingly the worst performer by Whisper and $n = 5$ which is the worst performer for Wav2Vec2 is the highest performer by Whisper. However, Whisper never goes under the baseline and is more consistent than Wav2vec2. The highest BLEU score reached is 29.47, which is approximately a 11% increase over the baseline, for $\alpha = 0.2$ with $n = 5$.

Looking at the means and standard deviations in Figure 5.7b, the method performs consistently over the baseline with similar oscillations. Any α value to be selected would perform well, however, there are small differences that can be considered. The $\alpha = 0.2$ has the best score within the selections but the average score of $\alpha = 0.2$ is slightly lower than $\alpha = 0.25$. The mean of $\alpha = 0.2$ is 28.222 with a standard deviation of 0.9, for $\alpha = 0.25$ the mean is slightly better with 28.286 and the standard deviation is lower with 0.76. This makes $\alpha = 0.25$ a more consistent and reliable selection.

Figure 5.8a showcases the locally normalized rescoring done by Whisper ASR. The performance of Whisper also with COMET is visibly better than Wav2Vec2's. The effect of increasing the n-best list size reduced the COMET score for this case and the scoring one



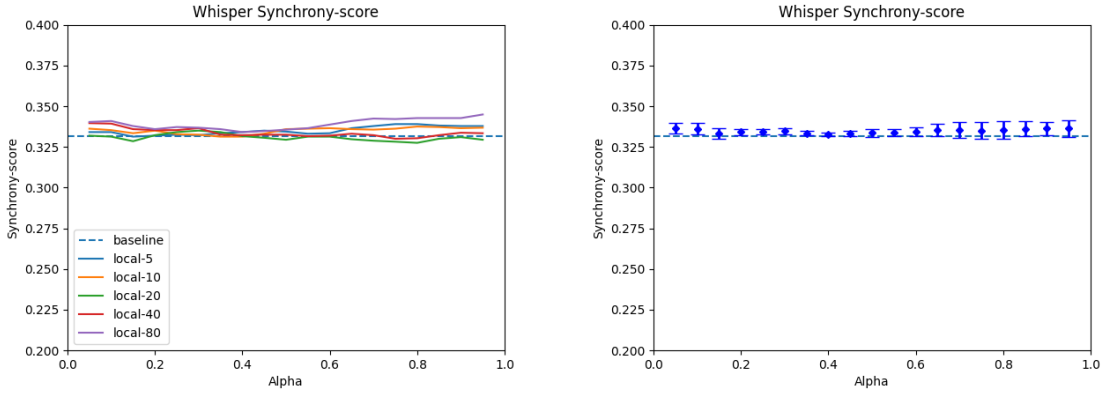
(a) COMET scores of N-best.

(b) Mean and standard deviation of N-bests.

Figure 5.8.: The distribution of COMET scores over the α values for rescoring done with locally normalized asr-scores.

is $n = 5$. All the lines in the graph are straight and horizontal meaning that the method is consistent for all the given α values. The highest score of 0.7975 is reached with $\alpha = 0.35$ and $n = 5$.

The mean and standard deviation graph in Figure 5.8b demonstrates that for each α value the scoring stays consistently better than the average. The mean difference between each point is small and can be overlooked. The same can also be said for the standard deviation changes between the α values. If the graph is deeply observed the highest performing average is at $\alpha = 0.3$ with 0.79048.



(a) Synchrony-scores of N-best.

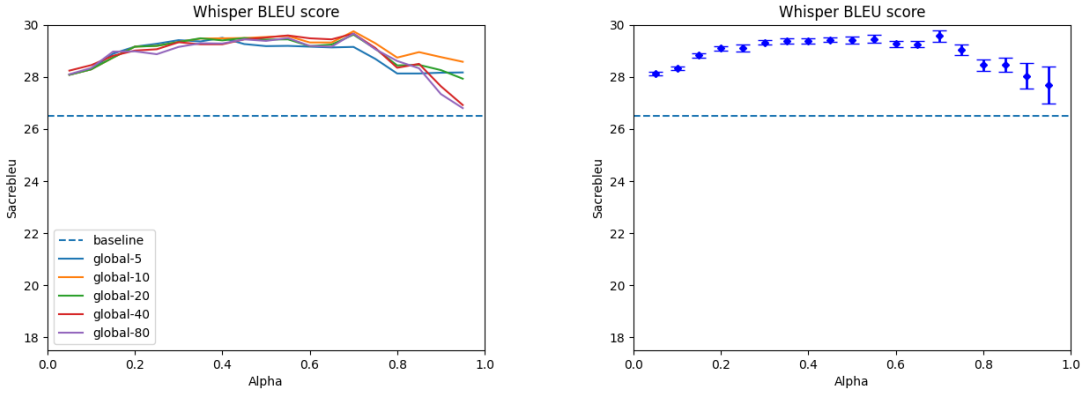
(b) Mean and standard deviation of N-bests.

Figure 5.9.: The distribution of synchrony-scores over the α values for rescoring done with locally normalized asr-scores.

Figure 5.9a showcases the synchrony-scores distribution over the α values for a locally normalized rescoring done with the Whisper ASR model. It is understandable from the graph that Whisper ASR does not optimize, its scoring on synchrony scores. It has an overall slight negative effect on the score and it only performs better than the baseline for

some cases. However, between different n-best list sizes, the best score 0.3275 belongs to the case $n = 20$ for $\alpha = 0.8$.

Further looking at Figure 5.9b, the similarity between the means can be observed. The mean of $\alpha = 0.2$ is 0.33338 is the lowest mean but the second lowest mean 0.33344 with $\alpha = 0.4$ has a lower standard deviation. It is hard to make a selection between both since the differences are so small that they can be ignored. Differences between the scores for any α value can be overlooked. Therefore making any selection based on synchrony-score for the locally normalized Whisper asr scoring is illogical and both average scores are worse than the baseline score 0.3316.



(a) BLEU scores of N-best.

(b) Mean and standard deviation of N-bests.

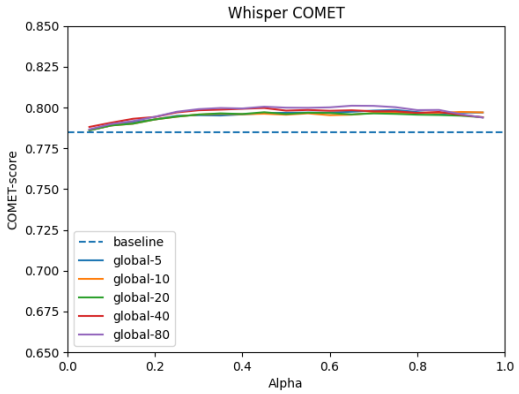
Figure 5.10.: The distribution of BLEU scores over the α values for rescoring done with globally normalized asr-scores.

Figure 5.10a represents the distribution of BLEU scores over the α values for different cases with various n-best list sizes for the globally normalized rescoring method done with the Whisper ASR model. All of the results are higher than the baseline. Compared to the locally normalized one the scores of cases are closer and from the first look none of the n-best list-sized cases sticks out. However, there is a clear peak of score at $alpha = 0.7$. Within all list sizes $n = 10$ performs the highest with a 29.75 BLEU score approximately a 12% increase over the baseline.

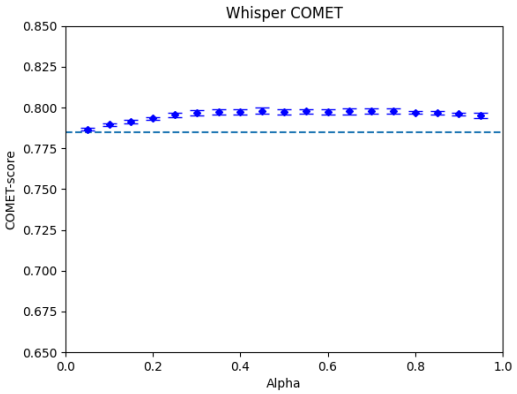
The second Figure 5.10b showcases the mean and standard deviation of BLEU scores over the α values. Interestingly higher α values have higher standard deviations. This graph also has the same peak as Figure 5.10a. The $\alpha = 0.7$ has the highest mean of 29.568 with a standard deviation of 0.21.

In Figure 5.11a the COMET values for the globally normalized scoring done with Whisper can be seen. Analogous to the locally normalized version all the cases perform better than the baseline. The lines in the graph are on top of each other meaning that the effect of different n-best list sizes is negligible. However, there is a subtle distinction where smaller n-best list sizes perform slightly worse, unlike the locally normalized scenario. The highest COMET score of 0.8011 is achieved by $\alpha = 0.65$ and $n = 80$.

Figure 5.11b illustrates the mean and standard deviation distribution of all n-best list sizes over the α values. A glance at the graph reveals that all standard deviations are very



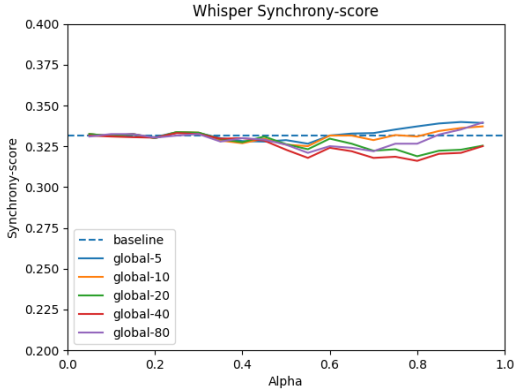
(a) COMET scores of N-best.



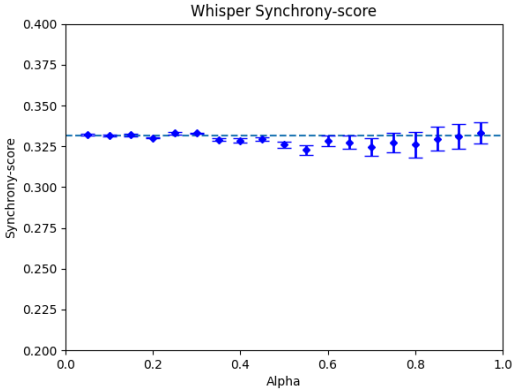
(b) Mean and standard deviation of N-bests.

Figure 5.11.: The distribution of COMET scores over the α values for rescoring done with globally normalized asr-scores.

small, indicating a consistent COMET score across various n-best list sizes. For every lower α value the method performs like the baseline model, however, at $\alpha = 0.45$ the average score peaks to 0.7980 averaging higher than locally normalized versions' peak COMET score, suggesting that the globally normalized version consistently outperforms its counterpart.



(a) Synchrony-scores of N-best.



(b) Mean and standard deviation of N-bests.

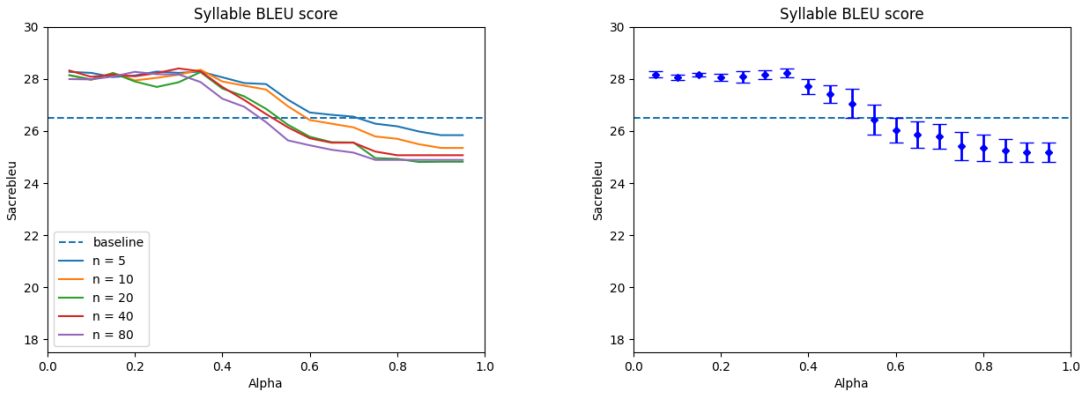
Figure 5.12.: The distribution of synchrony-scores over the α values for rescoring done with globally normalized asr-scores.

Figure 5.12a shows the synchrony-score distribution over α values for globally normalized rescoring done with the Whisper model. The cases show no differences till the point $\alpha = 0.5$. From there on the $n = 40$ case starts performing better among other cases and passes the baseline score. The $n = 5, 10$ cases perform similarly worse and $n = 20$ fluctuates and matches the score of $n = 40$ for high α values. The case $n = 80$ performs well between $0.5 < \alpha < 0.7$ and then starts falling behind. The best score achieved is 0.3161 by the case $n = 40$ with $\alpha = 0.8$.

Figure 5.12b showcases the mean and standard deviation of the synchrony scores over the α values. Looking at the mean values from the graph, three of the values stick out $\alpha = 0.55, 0.7, 0.8$. The $\alpha = 0.55$ has a mean of 0.3227 with a standard deviation of 0.0030, the $\alpha = 0.7$ has a mean of 0.3248 with a standard deviation of 0.0054, and the $\alpha = 0.8$ has a mean of 0.3259 with a standard deviation of 0.0077. Even though, $\alpha = 0.8$ has the best score among all the cases, $\alpha = 0.55$ returns on average better and more consistent results among all cases.

Whisper demonstrates substantial improvement with BLEU and COMET metric. In both local and global cases it shows over 10% increases over the baseline BLEU score. In contrast to the Wav2Vec2 model, the COMET score not only increases but also surpasses the baseline score, showcasing Whisper's capabilities in enhancing both dubbing and translation quality. For synchrony score, Whisper performs worse than the baseline. This can be interpret as Whispers incapability of learning the synchrony constraint. However, despite that having an increase in dubbing quality can imply that it might have learned a dubbing constraint, which have not been implicitly considered.

5.1.3. Syllable



(a) BLEU scores of N-best.

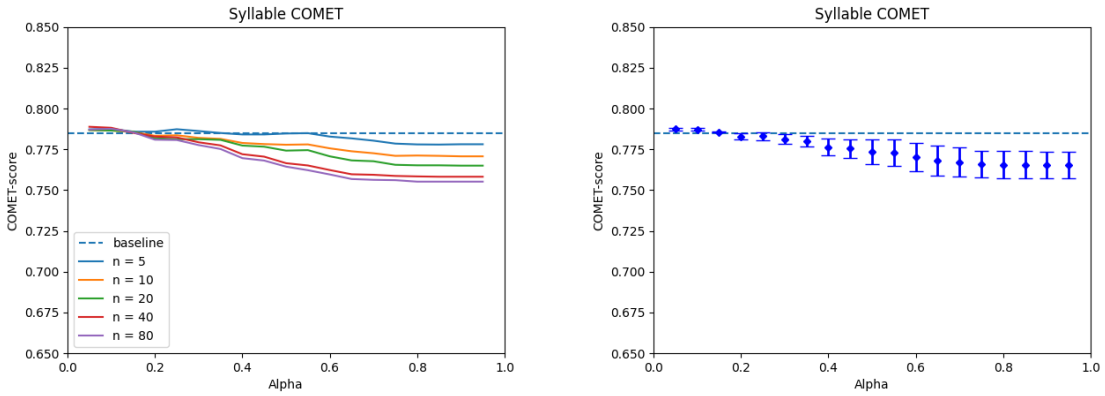
(b) Mean and standard deviation of N-bests.

Figure 5.13.: The distribution of BLEU scores over the α values for syllable rescoring.

The Figure 5.13a represents the syllable rescoring methods performance. The BLEU scores are higher than the baseline for cases $n = 5, 10$ with $\alpha < 0.6$ and for cases $n = 20, 40, 80$ with $\alpha < 0.5$. It can be directly seen that for $n = 5$ and $n = 10$, there is no disparity between the BLEU scores. The highest score achieved with this method is 28.40 with $\alpha = 0.3$ which is an approximately 7% increase over the baseline. The highest score is achieved by $n = 40$.

Figure 5.13b shows the mean and standard deviation of BLEU scores over the α values. Exactly at $\alpha = 0.55$, the average score produced by the method started to perform worse than the baseline and this continues for higher α values. Even though, the peak score of the method was achieved with $\alpha = 0.3$, $\alpha = 0.35$ provides a better mean of 28.212 with a

standard deviation of 0.16. For more consistent results for any n-best list size $\alpha = 0.35$ is a better choice.

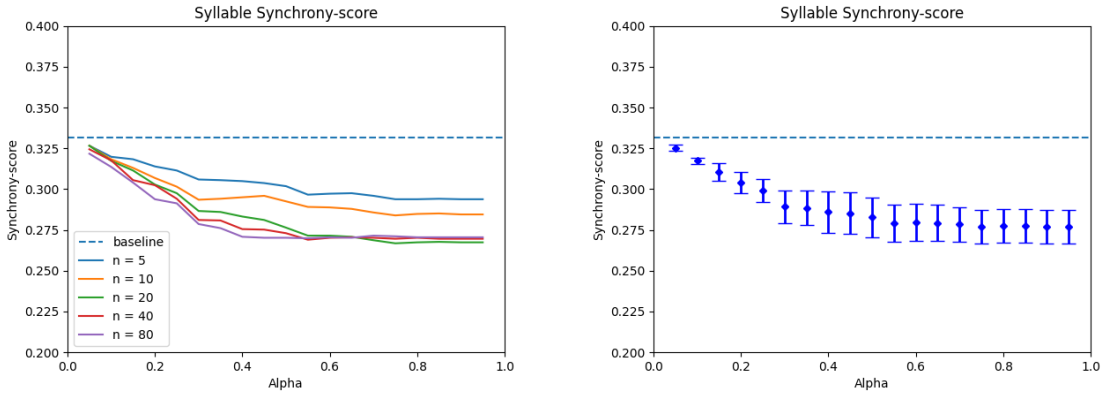


(a) COMET scores of N-best. (b) Mean and standard deviation of N-bests.

Figure 5.14.: The distribution of COMET scores over the α values for syllable rescoring.

Figure 5.14a illustrates the COMET score distribution over α values for all n-best list-sized cases. All the cases perform better than the baseline model for $\alpha < 0.2$. Notably, the line representing $n = 5$ exhibits greater consistency and experiences less performance loss for higher α values. For higher α values it becomes more visible that the increase of n-best list size affects the overall consistency of COMET scores. The highest score of 0.7888 is achieved with $\alpha = 0.05$ and $n = 40$. However, the gain is negligible since the baseline is 0.7851.

In Figure 5.14b showcases the mean and standard deviation of COMET scores over the α values. The graph exhibits a resemblance to the locally normalized Wav2Vec2 method, where the approach outperforms the baseline only for lower α values. Another parallel is the increase in standard deviation with rising α values. The method peaks 0.78734 for $\alpha = 0.05$.



(a) Synchrony-scores of N-best. (b) Mean and standard deviation of N-bests.

Figure 5.15.: The distribution of synchrony-scores over the α values for syllable rescoring.

Figure 5.15a of syllable rescoring shows a gain for synchrony-score. Increasing the α also increases the synchrony score for all cases. This makes sense since the syllable rescoring encourages the translation with syllable count closer to the source meaning that for $\alpha = 1$ the best possible selection for synchrony-score will be done. The best score is 0.2674 which is an approximately 20% decrease from the baseline, achieved by the case $n = 20$ with $\alpha = 0.9, 0.95$.

The second Figure 5.15b showcases the mean and standard deviation of synchrony scores over the α values. As expected the mean gets lower with increasing α values. The mean synchrony scores are always better than the baseline and it is expected for all the cases.

The syllable rescoring method showcases increase in BLEU score for certain α values. Similarly to the Wav2Vec2 model its COMET scores are not stable and is in a decreasing trend, showcasing its inability to preserve the translation quality. The synchrony score is always better than the baseline model. This is no surprise, after all the methods tends to select translations which have similar syllable counts to the source sentence.

5.1.4. Overview

The analysis for the selection of the hyperparameter α showed that each model had a different α value for its best BLEU, COMET, and synchrony score. The alignment scores calculated with Wav2Vec2 had a positive impact on the synchrony score with cases that have higher n-best list sizes. However, the alignment scores calculated with Whisper had little to no effect on that matter. Therefore Whisper ASR also proves that there is no direct relation between the synchrony score and the BLEU score. One can interpret that a good synchrony score does not always confirm the translation to be a good dubbing. For cases with Wav2Vec2 and Syllable rescoring increasing BLEU scores, the COMET score decreased. Showcasing the sacrifice in translation quality for gains in dubbing quality. However, Whisper preserves and improves the translation quality whilst increasing the dubbing quality.

Model	BLEU	COMET	Synchrony
Baseline	26.50	0.7851	0.2897
+Syllable$_{n=40,\alpha=0.3}$	28.40	0.7793	**0.2811**
+Wav2Vec2$_{local,n=40,\alpha=0.2}$	29.04	0.7789	0.3077
+Wav2Vec2$_{global,n=40,\alpha=0.35}$	28.92	0.7815	0.3167
+Whisper$_{local,n=5,\alpha=0.2}$	29.47	0.7956	0.3319
+Whisper$_{global,n=10,\alpha=0.7}$	**29.75**	**0.7967**	0.3288

Table 5.1.: The highest BLEU scores selected from all cases.

Table 5.1 displays the cases with the best BLEU scores for each method. The best-performing case and model overall are the globally normalized scores on `Whisper` with $n = 10$ and following that the locally normalized scores on the Whisper model with $n = 5$. For `Wav2Vec2`, both local and global normalization methods worked best with the $n = 40$. `Syllable` rescoring performed better than the baseline however it performed worse than

Model	BLEU	COMET	Synchrony
Baseline	26.50	0.7851	0.2897
+Syllable$_{n=40,\alpha=0.3}$	28.21	0.7800	**0.2885**
+Wav2Vec2$_{local,n=40,\alpha=0.2}$	28.61	0.7783	0.3117
+Wav2Vec2$_{global,n=40,\alpha=0.35}$	28.59	0.7777	0.3064
+Whisper$_{local,n=5,\alpha=0.2}$	28.28	0.7899	0.3343
+Whisper$_{global,n=10,\alpha=0.7}$	**29.56**	**0.7980**	0.3248

Table 5.2.: The highest average BLEU scores of all the n-best list sizes for all versions of the method.

the asr-scoring method. For `Wav2Vec2` and `Syllable` cases, it can be seen that an overall increase in the BLEU score is achieved, however, a similar gain cannot be observed for the COMET score. This can mean that even though the performance for dubbing increased, the translation quality decreased slightly. For `Whisper` cases an increase can be seen in both COMET and BLEU scores, showcasing methods superiority.

Table 5.2 displays the cases with the best average BLEU score over the n-best list sizes for each method. The α values of this table vary from the last table since not all the best BLEU yielding α values also perform overall the best. All methods provided in this table perform better than the baseline. The best method is the globally normalized rescoring done with `Whisper`, interestingly there is a high gap between locally and globally normalized versions. `Wav2Vec2` is the second-best performer and it is more consistent with both normalizations. The worst-performing method is the `Syllable` rescoring. For COMET scores only the `Whisper` model performs over the baseline and for synchrony-score only the `Syllable` model outperforms the baseline.

5.2. Evaluation of methods knowing speaker visibility

This section will go into the evaluation of each experiment case, with two primary scenarios under consideration. In the first case, both audio and corresponding transcribed text data are available and can be utilized, this scenario is denoted as MT_{en-es}. In the second case, it is assumed that only the audio data is accessible, necessitating the creation of transcriptions through ASR models, this scenario is denoted as $Whisper_{en}^{ASR} \rightarrow Mt_{en-es}$. For both scenarios, the same test datasets are employed and further categorized based on the visibility of the speaker. The test dataset is segmented into four sections: "on", "off", "mixed", and "all.". This segmentation of the data is more detailedly explained in Section 4.3.3.

For the following evaluation section, α values are selected based on the consistency of the BLEU scores, which is displayed in Table 5.2. The only n-best list size selected is $n = 10$ because the best performing method `Whisper`$_{global}$ showcased that $n = 10$ is large enough to produce competent results.

5.2.1. On

The "on" segment of the set test only includes the speech parts where the speaker is on screen and their mouth is visible. It is important that the translated versions also require similar mouth movements to be uttered so that lip synchrony can be ensured. Therefore the main evaluation of this method going to be done for this segment of the test set.

Table 5.3 showcases the BLEU and COMET scores of the rescoring methods for both scenarios. For the first scenario (MT_{en-es}), each rescoring method results in a positive BLEU gain over the baseline. The lowest of them being the `Syllable` rescoring method. Following that is the asr-rescoring done with `Wav2Vec2` model, for both normalized versions. The highest performer overall for this set is the asr-rescoring done with the $\texttt{Whisper}_{local}$ method. Providing a gain of 10% over the baseline BLEU score. Only the `Whisper` methods perform higher than the baseline for COMET scores. The highest performer for COMET is the $\texttt{Whisper}_{global}$ method.

For the second scenario ($Whisper_{en}^{ASR} \rightarrow Mt_{en-es}$), each method again results in positive gains over the baseline score. This time $\texttt{Wav2Vec2}_{global}$ has a slightly worse score than `Syllable` rescoring. Both Whisper methods again occupy the best two spots. However, this time $\texttt{Whisper}_{global}$ has a slight edge over the $\texttt{Whisper}_{local}$ with a BLEU score of 29.41 an approximate 12% increase over the baseline. For the COMET scores this scenario results similarly, with only the `Whisper` methods outperforming the baseline and $\texttt{Whisper}_{global}$ method being the best scorer.

Model	Mt_{en-es}		$Whisper_{en}^{ASR} \rightarrow Mt_{en-es}$	
	BLEU	COMET	BLEU	COMET
Baseline	29.85	0.7974	26.34	0.7729
+Syllable$_{\alpha=0.35}$	30.29 / +1.4%	0.7863 / -1.1%	27.92 / +5.9%	0.7669 / -0.6%
+Wav2Vec2$_{local,\alpha=0.2}$	31.31 / +5.2%	0.7903 / -0.7%	28.52 / +8.0%	0.7700 / -0.3%
+Wav2Vec2$_{global,\alpha=0.45}$	31.22 / +5.0%	0.7919 / -0.5%	27.88 / +5.7%	0.7706 / -0.2%
+Whisper$_{local,\alpha=0.25}$	**32.89 / +10.2%**	0.8082 / +1.3%	29.17 / +10.4%	0.7857 / +1.6%
+Whisper$_{global,\alpha=0.7}$	32.56 / +9.0%	**0.8095 / +1.6%**	**29.41 / +11.2%**	**0.7893 / +2.1%**

Table 5.3.: BLEU and COMET scores and the gains of the methods over the baseline model for the "on" segment of the test set.

5.2.2. Off

The "off" segment of the dataset includes the speech files where the speaker is not on screen. This segment should not be as constrained for dubbing as the "on" segment. However, it can still be constrained by time if the scene switches directly to someone else talking, or where the talking character's lips are visible in the next scene. Therefore there might be some positive effects of these methods.

Table 5.4 shows the BLEU and COMET scores of the rescoring methods for given scenarios. For the first scenario (MT_{en-es}), the baseline BLEU score is 32.65. Directly it can be observed that the MT systems output without any rescoring performs as well as the

Model	Mt_{en-es}		$Whisper_{en}^{ASR} \rightarrow Mt_{en-es}$	
	BLEU	COMET	BLEU	COMET
Baseline	32.65	0.7945	29.65	0.7696
+Syllable$_{\alpha=0.35}$	32.96 / +1.0%	0.7879 / -0.8%	29.92 / +0.9%	0.7642 / -0.7%
+Wav2Vec2$_{local,\alpha=0.2}$	32.13 / -1.9%	0.7863 / -1.0%	29.45 / -0.9%	0.7609 / -1.1%
+Wav2Vec2$_{global,\alpha=0.45}$	32.57 / -1.1%	0.7886 / -0.7%	29.47 / -1.0%	0.7591 / -1.4%
+Whisper$_{local,\alpha=0.25}$	**34.70 / +6.3%**	0.8022 / +1.0%	**31.78 / +7.1%**	0.7791 / +1.2%
+Whisper$_{global,\alpha=0.7}$	33.92 / +4.4%	**0.8034 / +1.1%**	30.96 / +3.9%	**0.7804 / +1.4%**

Table 5.4.: BLEU and COMET scores and the gains of the methods over the baseline model for the "off" segment of the test set.

output of the best rescoring methods on the "on" set. Interestingly both `Wav2Vec2` methods have negative gains on the baseline. For this segment `Syllable` rescoring performs better than them with an increase from the baseline. The best performers are the Whisper models with `Whisper`$_{local}$ having the highest BLEU score of 34.7 an approximate 6% increase from the baseline. For COMET scores, the `Wav2Vec2` and `Syllable` methods perform worse than the baseline. The best performing method is the `Whisper`$_{global}$.

In the second scenario ($Whisper_{en}^{ASR} \rightarrow Mt_{en-es}$), both of the Wav2Vec2 methods again have a slightly lower BLEU score from the baseline. `Syllable` rescoring method has a slight positive gain over the baseline. Again the highest scoring two methods are the Whisper methods and `Whisper`$_{local}$ is the highest scoring, having an increase around 7% BLEU score from the baseline. In terms of COMET scores, this situation is identical, where only the `Whisper` methods surpass the baseline, and the `Whisper`$_{globa}$ method emerges as the top performer.

The noticeable increases in BLEU scores for all models in the "on" section highlight a significant improvement. However, a similar increase is not observed in the "off" section. As discussed earlier "on" segment is more constrained then "off", therefore having increases specifically for "on" demonstrates that the models are leaning towards selecting the translation candidates that are more suitable for dubbing constraints. Which indicates the methods capability of learning these constraints.

5.2.3. Mixed

The "mixed" segment of the dataset consists of speech files from the scenes where the speaker is partially on screen and their mouth is visible. In these scenes, the rescoring methods are expected to perform well, as even though some parts of the scenes are off-screen, there are still time constraints imposed on them.

Table 5.5 showcases the BLEU and COMET scores of the rescoring methods for both scenarios for the mixed datasets. For the first scenario (Mt_{en-es}), all the methods perform similarly well with positive BLEU score gains over the baseline score. The methods `Syllable` rescoring, `Wav2Vec2`$_{local}$, and `Wav2Vec2`$_{global}$ have a gain of 7% over the baseline. The best performing methods are both `Whisper-local` and `Whisper`$_{global}$ methods with 9% and 8% gains over the baseline score, respectively. For this segment of the dataset, the

Model	Mt_{en-es}		$Whisper_{en}^{ASR} \rightarrow Mt_{en-es}$	
	BLEU	COMET	BLEU	COMET
Baseline	28.84	0.7676	27.08	0.7543
+Syllable$_{\alpha=0.35}$	30.91 / +7.2%	0.7621 / -0.7%	27.80 / +2.7%	0.7492 / -0.7%
+Wav2Vec2$_{local,\alpha=0.2}$	30.81 / +6.8%	0.7651 / -0.5%	28.63 / +5.7%	0.7523 / -0.3%
+Wav2Vec2$_{global,\alpha=0.45}$	30.96 / +7.4%	0.7655 / -0.2%	28.55 / +5.4%	0.7522 / -0.2%
+Whisper$_{local,\alpha=0.25}$	**31.36 / +8.7%**	**0.7846 / +2.9%**	**29.75 / +9.9%**	**0.7708 / +2.4%**
+Whisper$_{global,\alpha=0.7}$	31.17 / +8.1%	0.7832 / +2.5%	29.36 / +8.4%	**0.7708 / +2.4%**

Table 5.5.: BLEU and COMET score and the gains of the methods over baseline model for the "mixed" segment of the test set.

COMET scores of `Wav2Vec2` and `Syllable` methods are nearly identical to the baseline and both `Whisper` methods outperform the baseline and the `Whisper`$_{local}$ being the best performer.

For the second scenario ($Whisper_{en}^{ASR} \rightarrow Mt_{en-es}$), all the methods have a positive gain over the baseline. `Syllable-rescoring` method has the worst performance among the methods with a score of 27.80. Next in the ranking is both the Wav2Vec2 methods with the global one having a 28.55 and the local one having a slightly better score of 28.63. Among the Whisper methods, the `Whisper-local` performs the best with a score of 29.75 which is around a 10% increase over the baseline score. Similarly, the COMET scores for `Wav2Vec2` and `Syllable` methods closely resemble the baseline, while both `Whisper` methods surpass the baseline, with `Whisper`$_{local}$ emerging as the top performer.

5.2.4. All

The "all" segment contains an aggregation of "on", "off", and "mixed" dataset segments. Experimenting with the "all" segment is more realistic since doing a segmentation of the dataset for on and off-screen scenes is heavy work in preprocessing. The results for this segment may vary with the rate of "on" and "off" segments. For this study, the "all" segment includes an equal amount of sentences from the "on", "off" and "mixed" segments.

Model	Mt_{en-es}		$Whisper_{en}^{ASR} \rightarrow Mt_{en-es}$	
	BLEU	COMET	BLEU	COMET
Baseline	30.17	0.7865	27.67	0.7656
+Syllable$_{\alpha=0.35}$	31.43 / +4.2%	0.7786 / -2.2%	28.49 / +3.0%	0.7600 / -3.2%
+Wav2Vec2$_{local,\alpha=0.2}$	31.64 / +4.9%	0.7799 / -1.0%	28.93 / +4.6%	0.7595 / -0.3%
+Wav2Vec2$_{global,\alpha=0.45}$	31.65 / +4.9%	0.7817 / -0.7%	28.80 / +4.1%	0.7605 / -0.8%
+Whisper$_{local,\alpha=0.25}$	**32.33 / +7.2%**	0.7937 / +1.0%	**29.70 / +7.3%**	0.7739 / +0.8%
+Whisper$_{global,\alpha=0.7}$	32.04 / +6.2%	**0.7949 / +1.3%**	29.51 / +6.6%	**0.7755 / +1.0%**

Table 5.6.: BLEU and COMET score and the gains of the methods over the baseline model for the whole test set.

The BLEU and COMET scores of the rescoring methods for both scenarios, with the "all" segment, are presented in Table 5.6. In the first scenario (Mt_{en-es}), all the methods display comparable and favorable performance with positive gains for the BLUE score over the baseline. The `Syllable` rescoring method has the lowest gains among all methods with an approximate 4% increase over baseline. Both Wav2Vec2 methods provide a gain of 5%. The best performing methods are the Whisper methods with `Whisper-local` scoring the highest with 32.33 which is a 7% gain on the baseline score. The COMET scores of `Wav2Vec2` and `Syllable` methods are slightly lower than the baseline. Both `Whisper` methods outperform baseline with $\texttt{Whisper}_{global}$ being the top performer.

For the second scenario ($Whisper_{en}^{ASR} \rightarrow Mt_{en-es}$), all the methods have a positive gain over the baseline. The `Syllable-rescoring` method performs the poorest among all other methods. The Wav2Vec2 method achieves around a 4 to 5% increase on the baseline. The best-performing method is the `Whisper-local`. However, both Whisper methods perform similarly well and have a gain of around 7% over the baseline. Same as the first scenario the COMET scores for `Wav2Vec2` and `Syllable` methods are lower than the baseline, while both `Whisper` methods outperform the baseline, with $\texttt{Whisper}_{global}$ having the best score.

5.2.5. Synchrony-score

In this section, a brief analysis of synchrony scores is conducted. Since this metric represents a measurement for the dubbing constraints, these results will showcase how much each of the methods is learning the dubbing constraints.

Model	Mt_{en-es}			$Whisper_{en}^{ASR} \rightarrow Mt_{en-es}$		
	On	Off	Mixed	On	Off	Mixed
Baseline	0.3146	0.3669	0.2948	0.3134	0.3674	0.2896
+Syllable$_{\alpha=0.35}$	**0.2868**	**0.3368**	0.2728	**0.2836**	**0.3457**	**0.2696**
+Wav2Vec2$_{local,\alpha=0.2}$	0.2921	0.3561	0.2732	0.2931	0.3574	0.2747
+Wav2Vec2$_{global,\alpha=0.45}$	0.2940	0.3587	**0.2704**	0.2931	0.3574	0.2777
+Whisper$_{local,\alpha=0.25}$	0.3188	0.3752	0.2994	0.3240	0.3796	0.3004
+Whisper$_{global,\alpha=0.7}$	0.3260	0.3823	0.3008	0.3304	0.3850	0.3019

Table 5.7.: The synchrony-score of both scenarios with "on", "off" and "mixed" segments of the Heroes dataset.

Table 5.7 showcases the synchrony scores for each method on both scenarios with "on", "off" and "mixed" segments. Firstly all the methods other than the `Whisper` methods show positive gains over the baseline model. Generally, the best performing method is `Syllable` and it is expected since the method tries to optimize this metric on translation candidates. The `Wav2Vec2` models show gains for the synchrony-score showcasing that the method is helping the model learn the dubbing constraints. However, `Whisper` the best-performing method for BLEU and COMET scores performs always worse than the baseline for synchrony score.

The synchrony score only concerns itself with the synchrony constraint of dubbing. It is clear that both `Syllable` and `Wav2Vec2` has adapted to this constraint and increased

the dubbing quality for certain cases. However, the best scoring model `Whisper` has done poorly on adapting itself to this constraint. The performance increase it provided in dubbing, is then not tied to this constraint.

This raises a question: For what constraints does Whisper adapt itself? The discrepancy in performance metrics suggests that Whisper may implicitly learn and incorporate constraints not explicitly considered.

5.2.6. Overview

Looking at the scores from a speaker visibility perspective, it can be said that for the "on" scenes where the speaker is visible and the dubbing constraints are at most, the proposed asr-scoring method outperformed the baseline in terms of the BLEU score. Both versions of the method show improvements over the `Syllable` rescoring method. This improvement in the "on" segment shows great promise in the methods' capability of adapting the model to the dubbing constraints.

Contrastingly, evaluating the method's performance on the "off" set, where speakers are not visible, and dubbing constraints are minimal, places a higher emphasis on translation quality than dubbing quality. For this set the translation quality is more significant than the dubbing quality. It can be observed that the rescoring method with the `Wav2Vec2` model performs slightly worse than the baseline, indicating that the method is not optimizing itself for the translation quality, however, the method with `Whisper` shows some improvements on the set by BLEU and by COMET score. The improvements are not significantly high but demonstrate positive gains in a set where it is expected to perform poorly, highlighting the method's strength in preserving the translation quality.

The results on the "all" set demonstrate the capabilities of the method in a scenario where a set separation by visibility of the speaker is not done. Both versions of the proposed method performed better than the baseline and the `Syllable` rescoring method. Notably, the `Whisper` method stands out as the clear outperformer with BLEU and COMET scores. Improvements on this set are very crucial since splitting the dataset into on and off-screen might not be always easily achievable, and highlights the method's capability of working even in scenarios that are explicitly not optimized for itself.

Finally, it can be said that each method makes a trade-off between dubbing quality and translation quality whilst selecting the α values. Therefore it is acceptable to sacrifice the translation quality to have improvements for the dubbing. For the COMET metric, the only methods that outperformed the baseline among all cases are the `Whisper` methods. Despite all the α values for the methods being selected based on their dubbing performance on the "on" validation set, `Whisper` preserves and improves the translation quality. This resilience and improvement in translation quality underscore the robustness of the `Whisper` method.

6. Conclusion

This Chapter concludes the thesis by summarizing it in Section 6.1, giving answers to the research questions in Section 6.2, and discussing improvements for the study in Section 6.3.

6.1. Summary

This thesis proposes a novel rescoring method that utilizes audio features to improve the performance of MT models for dubbing scenarios. In contrast to previous rescoring methods [10, 36], which built their baseline models by training vanilla transformer models from scratch, this study employs a pre-trained model, DeltaLM, fine-tuned specifically for the dubbing process as the baseline. The baseline model exhibits higher-quality translations compared to previous studies, serving as a benchmark for evaluating the effectiveness of newer methods against former ones.

The training set consists of OpenSubtitles [42] a massive dataset of parallel subtitles, and Heroes [51], a small dubbing dataset. To create a scoring that showcases the audio features' importance for the translation candidates the forced alignment technique is conducted. This technique involves aligning a given audio file's time-frames to preselected words and providing probabilities of each word being aligned to the given time-frame. These probabilities are then utilized for rescoring and to have a more comprehensive understanding of the new rescoring technique, two different ASR models Wav2Vec2 [7] and Whisper [58] are selected. Further observing with both models showcased great differences between them indicating the importance of the quality of the ASR model. Additionally, a new alignment decoder for the Whisper Asr model is developed. This decoder allows the Whisper ASR model to be used for forced alignment tasks.

For evaluation, first, a hyperparameter analysis is performed to select a hyperparameter to find an optimal weight for the forced decoding score. The method is then tested on test sets that are segmented by the speaker visibility [36]. This segmentation provided insights into the methods' performance across various scenarios with differing dubbing constraints. Overall, the approach introduces a valuable contribution to the field by combining pre-trained models, forced alignment, and ASR models to optimize the translation process for dubbing applications.

6.2. Answers to Research Questions

The thesis mainly revolves around two research questions that were given in Section 1.2. The evaluation provides answers to these questions.

RQ1: Can the dubbing constraints be automatically learned?

The results presented in Section 5.2.1 highlight the performance of the novel method on both ASR models. It can be seen that for both scenarios the new method performed better than the baseline and previous methods for the "on" set. It can not be denied the methods' capability of improving the dubbing quality. However, in Section 5.2.5 the synchrony scores for the method is shown. This metric can be interpreted as the synchrony constraint of dubbing. Notably, the method utilizing `Wav2Vec2` model demonstrates gains in synchrony scores, indicating the model's ability to learn and adhere to synchrony constraints inherent in dubbing.

On the other hand, the method with `Whisper` model, performs worse than the baseline for this given metric but with overall the most gains for the dubbing quality. For the method with `Wav2Vec2`, the automatic learning of constraints is apparent. In contrast, for the method with `Whisper`, while a direct conclusion about learning specific constraints may not be drawn, the observed improvement in the context of dubbing suggests the potential for implicit learning of constraints that might not be explicitly considered.

RQ2: How does adapting to dubbing constraints affect the translation quality?

The second question concerns itself more about the translation quality. At the start of Chapter 5 each metric and their resemblance is discussed. The metric that focuses the most on translation quality is the COMET metric. Further in evaluation, it was seen that for `Syllable` rescoring and the method with `Wav2Vec2` model showed slight decreases in COMET comparison to the baseline. However, the method with `Whisper` demonstrated improvements over the baseline for all the cases. For some of the cases the cases the increase was negligible. The answer to this question depends on the ASR model that is used for the method. While `Wav2Vec2` had a slight decrease in translation quality, `Whisper` had a positive effect.

6.3. Future Work

This study provided a novel rescoring method that provided improvements for dubbing scenarios. For future work, new approaches with similar ideas can be explored or improvements upon this work can be made. Since this work has shown that the audio features do improve the dubbing quality an end-to-end model can be built and trained that can automatically learn the dubbing constraints. For this model, the encoder of Whisper can be used, since it showed great potential in this study.

To leverage the outcomes of this study, an auto dataset segmenter based on the visual appearance of the speaker can be developed. Such an application would automatically split the "on" and "off" segments of the dataset so that the use of this method could be optimized.

A similar rescoring can be done with the newly developed audio-visual model MuaViC [4]. This way the dubbing constraints can be learned from both audio and visual features.

Bibliography

[1] Mar. 2023. URL: https://www.statista.com/statistics/611707/online-video-time-spent/.

[2] Oct. 2023. URL: https://en.wikipedia.org/wiki/List_of_most-subscribed_YouTube_channels.

[3] Triantafyllos Afouras, Joon Son Chung, and Andrew Zisserman. "Deep Lip Reading: a comparison of models and an online application". In: *CoRR* abs/1806.06053 (2018). arXiv: 1806.06053. URL: http://arxiv.org/abs/1806.06053.

[4] Mohamed Anwar et al. *MuAViC: A Multilingual Audio-Visual Corpus for Robust Speech Recognition and Robust Speech-to-Text Translation*. 2023. arXiv: 2303.00628 [cs.CL].

[5] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. *Layer Normalization*. 2016. arXiv: 1607.06450 [stat.ML].

[6] Alexei Baevski, Michael Auli, and Abdelrahman Mohamed. "Effectiveness of self-supervised pre-training for speech recognition". In: *CoRR* abs/1911.03912 (2019). arXiv: 1911.03912. URL: http://arxiv.org/abs/1911.03912.

[7] Alexei Baevski et al. "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations". In: *CoRR* abs/2006.11477 (2020). arXiv: 2006.11477. URL: https://arxiv.org/abs/2006.11477.

[8] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *conference paper at ICLR 2015* (2015). URL: https://arxiv.org/abs/1409.0473.

[9] Max Bain et al. *WhisperX: Time-Accurate Speech Transcription of Long-Form Audio*. 2023. arXiv: 2303.00747 [cs.SD].

[10] Leclercq Bertrand. 2021. URL: https://dke.maastrichtuniversity.nl/jan.niehues/wp-content/uploads/2021/07/Leclerq-thesis.pdf.

[11] Ondřej Bojar et al. "Findings of the 2013 Workshop on Statistical Machine Translation". In: *Proceedings of the Eighth Workshop on Statistical Machine Translation*. Ed. by Ondrej Bojar et al. Sofia, Bulgaria: Association for Computational Linguistics, Aug. 2013, pp. 1–44. URL: https://aclanthology.org/W13-2201.

[12] Mauro Cettolo, C. Girardi, and Marcello Federico. "Wit3: Web inventory of transcribed and translated talks". In: *Proceedings of EAMT* (Jan. 2012), pp. 261–268.

[13] Caroline Chan et al. "Everybody Dance Now". In: *CoRR* abs/1808.07371 (2018). arXiv: 1808.07371. URL: http://arxiv.org/abs/1808.07371.

[14] KyungHyun Cho et al. "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches". In: *CoRR* abs/1409.1259 (2014). arXiv: 1409.1259. URL: http://arxiv.org/abs/1409.1259.

[15] Jan Chorowski et al. "Unsupervised speech representation learning using WaveNet autoencoders". In: *CoRR* abs/1901.08810 (2019). arXiv: 1901.08810. URL: http://arxiv.org/abs/1901.08810.

[16] J. Clement. *Twitch most popular language 2023*. July 2023. URL: https://www.statista.com/statistics/1133027/llanguage-twitch/.

[17] Ramandeep Singh Dehal et al. "GPU Computing Revolution: CUDA". In: *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*. 2018, pp. 197–201. DOI: 10.1109/ICACCCN.2018.8748495.

[18] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL].

[19] Mattia A. Di Gangi et al. "MuST-C: a Multilingual Speech Translation Corpus". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 2012–2017. DOI: 10.18653/v1/N19-1202. URL: https://aclanthology.org/N19-1202.

[20] Linhao Dong, Shuang Xu, and Bo Xu. "Speech-Transformer: A No-Recurrence Sequence-to-Sequence Model for Speech Recognition". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018, pp. 5884–5888. DOI: 10.1109/ICASSP.2018.8462506.

[21] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *Journal of Machine Learning Research* 12.61 (2011), pp. 2121–2159. URL: http://jmlr.org/papers/v12/duchi11a.html.

[22] Einar H. Dyvik. *The most spoken languages worldwide 2023*. June 2023. URL: https://www.statista.com/statistics/266808/the-most-spoken-languages-worldwide/.

[23] Thierry Etchegoyhen et al. "Machine Translation for Subtitling: A Large-Scale Evaluation". In: *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. Ed. by Nicoletta Calzolari et al. Reykjavik, Iceland: European Language Resources Association (ELRA), May 2014, pp. 46–53. URL: http://www.lrec-conf.org/proceedings/lrec2014/pdf/463_Paper.pdf.

[24] Souheil Fenghour et al. "Deep Learning-Based Automated Lip-Reading: A Survey". In: *IEEE Access* 9 (Aug. 2021), pp. 121184–121205. DOI: 10.1109/ACCESS.2021.3107946.

[25] Frederic B. Fitch. "Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. Bulletin of mathematical biophysics, vol. 5 (1943), pp. 115–133." In: *The Journal of Symbolic Logic* 9.2 (1944), pp. 49–50. DOI: 10.2307/2268029.

[26] Behrooz Ghorbani et al. “Scaling Laws for Neural Machine Translation”. In: *CoRR* abs/2109.07740 (2021). arXiv: 2109.07740. URL: https://arxiv.org/abs/2109.07740.

[27] Alex Graves et al. “Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural ’networks”. In: vol. 2006. Jan. 2006, pp. 369–376. DOI: 10.1145/1143844.1143891.

[28] Dan Hendrycks and Kevin Gimpel. “Bridging Nonlinearities and Stochastic Regularizers with Gaussian Error Linear Units”. In: *CoRR* abs/1606.08415 (2016). arXiv: 1606.08415. URL: http://arxiv.org/abs/1606.08415.

[29] Moto Hira. *Forced alignment with WAV2VEC2¶*. URL: https://pytorch.org/audio/main/tutorial/forced_alignment_tutorial.html.

[30] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-term Memory”. In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[31] Eric Jang, Shixiang Gu, and Ben Poole. *Categorical Reparameterization with Gumbel-Softmax*. 2017. arXiv: 1611.01144 [stat.ML].

[32] Ye Jia et al. “Direct speech-to-speech translation with a sequence-to-sequence model”. In: *CoRR* abs/1904.06037 (2019). arXiv: 1904.06037. URL: http://arxiv.org/abs/1904.06037.

[33] Dongwei Jiang et al. “Improving Transformer-based Speech Recognition Using Unsupervised Pre-training”. In: *CoRR* abs/1910.09932 (2019). arXiv: 1910.09932. URL: http://arxiv.org/abs/1910.09932.

[34] Prajwal K R et al. “Towards Automatic Face-to-Face Translation”. In: *Proceedings of the 27th ACM International Conference on Multimedia*. MM ’19. Nice, France: ACM, 2019. ISBN: 978-1-4503-6889-6. DOI: 10.1145/3343031.3351066. URL: http://doi.acm.org/10.1145/3343031.3351066.

[35] Nal Kalchbrenner and Phil Blunsom. “Recurrent Continuous Translation Models”. In: *Conference on Empirical Methods in Natural Language Processing*. 2013. URL: https://api.semanticscholar.org/CorpusID:12639289.

[36] Alina Karakanta et al. “The Two Shades of Dubbing in Neural Machine Translation”. In: *Proceedings of the 28th International Conference on Computational Linguistics*. Ed. by Donia Scott, Nuria Bel, and Chengqing Zong. Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 4327–4333. DOI: 10.18653/v1/2020.coling-main.382. URL: https://aclanthology.org/2020.coling-main.382.

[37] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[38] Alexander Kolesnikov et al. “Large Scale Learning of General Visual Representations for Transfer”. In: *CoRR* abs/1912.11370 (2019). arXiv: 1912.11370. URL: http://arxiv.org/abs/1912.11370.

[39] Taku Kudo and John Richardson. "SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing". In: *CoRR* abs/1808.06226 (2018). arXiv: 1808.06226. URL: http://arxiv.org/abs/1808.06226.

[40] Surafel Melaku Lakew et al. "Isometric MT: Neural Machine Translation for Automatic Dubbing". In: *CoRR* abs/2112.08682 (2021). arXiv: 2112.08682. URL: https://arxiv.org/abs/2112.08682.

[41] Surafel Melaku Lakew et al. "Machine Translation Verbosity Control for Automatic Dubbing". In: *CoRR* abs/2110.03847 (2021). arXiv: 2110.03847. URL: https://arxiv.org/abs/2110.03847.

[42] Pierre Lison and Jörg Tiedemann. "OpenSubtitles2016: Extracting Large Parallel Corpora from Movie and TV Subtitles". In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. Ed. by Nicoletta Calzolari et al. Portorož, Slovenia: European Language Resources Association (ELRA), May 2016, pp. 923–929. URL: https://aclanthology.org/L16-1147.

[43] Alexander H. Liu et al. "Towards Unsupervised Speech Recognition and Synthesis with Quantized Speech Representation Learning". In: *CoRR* abs/1910.12729 (2019). arXiv: 1910.12729. URL: http://arxiv.org/abs/1910.12729.

[44] Ilya Loshchilov and Frank Hutter. "Fixing Weight Decay Regularization in Adam". In: *CoRR* abs/1711.05101 (2017). arXiv: 1711.05101. URL: http://arxiv.org/abs/1711.05101.

[45] Shuming Ma et al. "DeltaLM: Encoder-Decoder Pre-training for Language Generation and Translation by Augmenting Pretrained Multilingual Encoders". In: *CoRR* abs/2106.13736 (2021). arXiv: 2106.13736. URL: https://arxiv.org/abs/2106.13736.

[46] Evgeny Matusov, Patrick Wilken, and Yota Georgakopoulou. "Customizing Neural Machine Translation for Subtitling". In: *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*. Ed. by Ondřej Bojar et al. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 82–93. DOI: 10.18653/v1/W19-5209. URL: https://aclanthology.org/W19-5209.

[47] Mjpost. *Mjpost/Sacrebleu: Reference Bleu implementation that auto-downloads test sets and reports a version string to facilitate cross-lab comparisons*. URL: https://github.com/mjpost/sacrebleu.

[48] Abdelrahman Mohamed, Dmytro Okhonko, and Luke Zettlemoyer. "Transformers with convolutional context for ASR". In: *CoRR* abs/1904.11660 (2019). arXiv: 1904.11660. URL: http://arxiv.org/abs/1904.11660.

[49] Khanh Nguyen and Hal Daumé III. "Global Voices: Crossing Borders in Automatic News Summarization". In: *CoRR* abs/1910.00421 (2019). arXiv: 1910.00421. URL: http://arxiv.org/abs/1910.00421.

[50] Keiron O'Shea and Ryan Nash. "An Introduction to Convolutional Neural Networks". In: *CoRR* abs/1511.08458 (2015). arXiv: 1511.08458. URL: http://arxiv.org/abs/1511.08458.

[51] Alp Oktem, Mireia Farrús, and Antonio Bonafonte. “Bilingual Prosodic Dataset Compilation for Spoken Language Translation”. In: Nov. 2018, pp. 20–24. DOI: 10.21437/IberSPEECH.2018-5.

[52] Myle Ott et al. “fairseq: A Fast, Extensible Toolkit for Sequence Modeling”. In: *Proceedings of NAACL-HLT 2019: Demonstrations*. 2019.

[53] Vassil Panayotov et al. “Librispeech: An ASR corpus based on public domain audio books”. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2015, pp. 5206–5210. DOI: 10.1109/ICASSP.2015.7178964.

[54] Kishore Papineni et al. 2002. URL: https://aclanthology.org/P02-1040.pdf.

[55] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *CoRR* abs/1912.01703 (2019). arXiv: 1912.01703. URL: http://arxiv.org/abs/1912.01703.

[56] Walter Pitts and Warren S. McCulloch. “How we know universals the perception of auditory and visual forms”. In: *The Bulletin of Mathematical Biophysics* 9.3 (1947), pp. 127–147. DOI: 10.1007/bf02478291.

[57] K. R. Prajwal et al. “A Lip Sync Expert Is All You Need for Speech to Lip Generation In The Wild”. In: *CoRR* abs/2008.10010 (2020). arXiv: 2008.10010. URL: https://arxiv.org/abs/2008.10010.

[58] Alec Radford et al. *Robust Speech Recognition via Large-Scale Weak Supervision*. 2022. arXiv: 2212.04356 [eess.AS].

[59] Zhiqiang Rao et al. “Length-Aware NMT and Adaptive Duration for Automatic Dubbing”. In: *Proceedings of the 20th International Conference on Spoken Language Translation (IWSLT 2023)*. Ed. by Elizabeth Salesky, Marcello Federico, and Marine Carpuat. Toronto, Canada (in-person and online): Association for Computational Linguistics, July 2023, pp. 138–143. DOI: 10.18653/v1/2023.iwslt-1.9. URL: https://aclanthology.org/2023.iwslt-1.9.

[60] Ricardo Rei et al. “COMET: A Neural Framework for MT Evaluation”. In: *CoRR* abs/2009.09025 (2020). arXiv: 2009.09025. URL: https://arxiv.org/abs/2009.09025.

[61] Olga Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *CoRR* abs/1409.0575 (2014). arXiv: 1409.0575. URL: http://arxiv.org/abs/1409.0575.

[62] Ashutosh Saboo and Timo Baumann. “Integration of Dubbing Constraints into Machine Translation”. In: *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*. Ed. by Ondřej Bojar et al. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 94–101. DOI: 10.18653/v1/W19-5210. URL: https://aclanthology.org/W19-5210.

[63] Steffen Schneider et al. “wav2vec: Unsupervised Pre-training for Speech Recognition”. In: *CoRR* abs/1904.05862 (2019). arXiv: 1904.05862. URL: http://arxiv.org/abs/1904.05862.

[64] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. "Sequence to Sequence Learning with Neural Networks". In: *CoRR* abs/1409.3215 (2014). arXiv: 1409.3215. URL: http://arxiv.org/abs/1409.3215.

[65] Jörg Tiedemann. "Parallel Data, Tools and Interfaces in OPUS". In: *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*. Ed. by Nicoletta Calzolari et al. Istanbul, Turkey: European Language Resources Association (ELRA), May 2012, pp. 2214–2218. URL: http://www.lrec-conf.org/proceedings/lrec2012/pdf/463_Paper.pdf.

[66] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[67] Martin Volk. "The Automatic Translation of Film Subtitles. A Machine Translation Success Story?" In: *J. Lang. Technol. Comput. Linguistics* 24 (2008), pp. 115–128. URL: https://api.semanticscholar.org/CorpusID:5186546.

[68] Martin Volk et al. "Machine Translation of TV Subtitles for Large Scale Production". In: (Nov. 2010). DOI: 10.5167/uzh-36755.

[69] Weiran Wang, Qingming Tang, and Karen Livescu. *Unsupervised Pre-training of Bidirectional Speech Encoders via Masked Reconstruction*. 2020. arXiv: 2001.10603 [eess.AS].

[70] Yonghui Wu et al. "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation". In: *CoRR* abs/1609.08144 (2016). arXiv: 1609.08144. URL: http://arxiv.org/abs/1609.08144.

[71] Egor Zakharov et al. "Few-Shot Adversarial Learning of Realistic Neural Talking Head Models". In: *CoRR* abs/1905.08233 (2019). arXiv: 1905.08233. URL: http://arxiv.org/abs/1905.08233.

[72] Thomas Zenkel et al. "Open Source Toolkit for Speech to Text Translation". In: *The Prague Bulletin of Mathematical Linguistics* 111 (Oct. 2018), pp. 125–135. DOI: 10.2478/pralin-2018-0011.

A. Appendix

A.1. Coding of the Alignment Decoder Module

```
1  class DecodingOptions:
2      task: str = "transcribe"
3      language: Optional[str] = None
4      temperature: float = 0.0
5      sample_len: Optional[int] = None
6      best_of: Optional[int] = None
7      beam_size: Optional[int] = None
8      patience: Optional[float] = None
9      length_penalty: Optional[float] = None
10     prompt: Optional[Union[str, List[int]]] = None
11     prefix: Optional[Union[str, List[int]]] = None
12     alignment_text: Optional[Union[str, List[int]]] = None
13     suppress_tokens: Optional[Union[str, Iterable[int]]] = "-1"
14     suppress_blank: bool = True
15     without_timestamps: bool = False
16     max_initial_timestamp: Optional[float] = 1.0
17     fp16: bool = True
```

Listing A.1: Decoding options of Whisper.

Listing A.1 shows the decoding options of Whisper that can be set by the user to specify the decoding process. A new parameter is added to the decoding options in line 12 to set up forced alignment. By setting the "alignment_text" to a string value, the user selects the option of doing forced alignment. That distinguishment is done when Whisper selects the decoder it is going to use for decoding.

In Listing A.2, the decoder selection can be seen. The priority is with the `BeamSearchDecoder`. If a beam_size is specified in the decoding options, the model prioritizes the use of a beam decoder. The second priority is with the `AlignmentDecoder`. Specification of alignment_text in decoding options forces the system to go into decoding with the alignment decoder. Lastly, if beam_size or alignment_text parameters are not specified, the model goes into decoding with `GreedyDecoder`.

Each decoder requires specific parameters. Both the beam and greedy decoder takes the `tokenizer.eot` parameter which is the `<|endoftranscript|>` token as input.

```
# decoder: implements how to select the next tokens,
# given the autoregressive distribution
if options.beam_size is not None:
    self.decoder = BeamSearchDecoder(
                    options.beam_size, tokenizer.eot,
                    self.inference, options.patience)
elif options.alignment_text is not None:
    self.decoder = AlignmentDecoder(
                    options.alignment_text,
                    tokenizer)
else:
    self.decoder = GreedyDecoder(options.temperature, tokenizer.eot)
```

Listing A.2: Selection of decoder.

Additionally, the beam decoder requires `beam_size`, `inference` and `patience`. The `patience` can be set by the user if not it defaults to 1.0. For greedy decoding the user can set `temperature` parameter, if it is not set it is defaulted to 0.0. The alignment decoder requires the `alignment_text` and the tokenizer. It does not take the `tokenizer.eot` since the tokenizer is passed along.

The decoders have three key functions. The `__init__()` function initializes the decoder and sets required local parameters. The `update()` function is where each quantized audio segment is processed. Lastly the `finalize()` function which concludes the decoders works for the given `decode()` command.

```
def __init__(self,alignment_text: List[str], tokenizer):
        self.eot = tokenizer.eot
        self.alignment_tokens = tokenizer.encode(
                                " " + alignment_text.strip())
        self.tokenizer = tokenizer
```

Listing A.3: The `__init__()` function of the `AlignmentDecoder`.

In Listing A.3 the `__init__()` function can be seen. It initializes all the local parameters to be used during the decoding process. The `alignment_text` is tokenized word by word and stored in a list for further use.

The `update()` function of a decoder goes through each quantized audio segment that is processed by the Whisper and assigns a most probable token. The `update()` function of the `AlignmentDecoder` assigns predefined word tokens to the audio segments that are iterated through Whisper. The `update()` function of the `AlignmentDecoder` can be seen in the Listing A.4. For input parameters, the function requires a set of `tokens`, which are the

```python
def update(
self, tokens: Tensor, logits: Tensor, sum_logprobs: Tensor,
) -> Tuple[Tensor, bool]:

    logprobs = F.log_softmax(logits.float(), dim=-1)
    next_tokens = logits.argmax(dim=-1)

    if self.tokenizer.decode(next_tokens) == "":
        pass
    elif not self.alignment_tokens:
        next_tokens[0] = torch.tensor([self.eot])
    else:
        next_tokens[0] = torch.tensor([self.alignment_tokens[0]])
        self.alignment_tokens.pop(0)

    current_logprobs = logprobs[torch.arange(logprobs.shape[0])
                                , next_tokens]
    sum_logprobs += current_logprobs * (tokens[:, -1] != self.eot)
    next_tokens[tokens[:, -1] == self.eot] = self.eot

    tokens = torch.cat([tokens, next_tokens[:, None]], dim=-1)
    completed = (tokens[:, -1] == self.eot).all()
    return tokens, completed
```

Listing A.4: The `update()` function of the `AlignmentDecoder`.

tokens that have been found through decoding or are predefined, a logits matrix which represents a classification of the audio segments on a classification of tokens, and the sum of the log-probabilities of the tokens that have been selected till now. The function starts by extracting a log-probabilities matrix from the logits matrix of Whisper. Then the `next_token` is set with the highest probability token from the logits matrix. Then it is controlled if the selected token is blank. If it is a blank token then the current update state is skipped, if not the predefined token list is checked to see if there are still existing tokens to be forcefully aligned. If the list is empty, `next_token` is set to the `<|endoftranscript|>` token. If the list is not empty, `next_token` is set to the first token from the list and the first element is popped. Now that the `next_token` is set the `current_logprobs` of that token is calculated and then added to the `sum_logprobs`. The `next_token` is then concatenated to the `tokens` list and then the function returns the new tokens and a value `complete` which denotes if the tokens set has a `<|endoftranscript|>`.

The last function `finalize()` of the `AlignmentDecoder` can be seen in the Listing A.5. This function is not modified and is the same as the `finalize()` function of the `Greedy-`

```
def finalize(
    self, tokens: Tensor, sum_logprobs: Tensor
) -> Tuple[Sequence[Sequence[Tensor]], List[List[float]]]:
    tokens = F.pad(tokens, (0, 1), value=self.eot)
    return tokens, sum_logprobs.tolist()
```

Listing A.5: The `finalize()` function of the `AlignmentDecoder`.

`Decoder`. This function makes sure each sequence has at least one `<|endoftranscript|>` token at the end.