EMBEDDED AND DISTRIBUTED AI SPEECH TRANSLATION PROJECT

Jesper Isacson
Computer Science
Jönköping University
Jönköping, Sweden
`isul18wj@student.ju.se`

Jorge Aguirregomezcorta Aina
Computer Science
Jönköping University
Jönköping, Sweden
`agjo22mm@student.ju.se`

Oskar Pettersson
Computer Science
Jönköping University
Jönköping, Sweden
`peos22wz@student.ju.se`

Samuel Mcmurray
Computer Science
Jönköping University
Jönköping, Sweden
`mcsa22oo@student.ju.se`

December 22, 2023

ABSTRACT

With a growing need for individuals to communicate with others in an increasingly connected world, the need for machine translation devices is more important than ever. This project implements a speech translation embedded system capable of capturing audio data and outputting text in another language. The implemented system consists of a multi-module sequential architecture that divides the operating requirements into individual, autonomous blocks. The different modules are a multi-language speech-to-text model, a multi-language text-translation system, and a smart-reply model. The speech translation is capable of working with three languages (English, Spanish, and Swedish) in an embedded environment, successfully implementing the base requirements of the project.

Github link: https://github.com/Keno-git/Embedded-Project

1 Introduction

In an age where international travel is a very common occurrence, many people rely on their English-speaking skills to maintain conversations with the native people from the country they visit. However, there are still many countries that have not established English as a second or third language, or even people who have difficulties developing their language skills but still want to connect with other parts of the globe. For such reasons, many people have to rely on technology to satisfy their needs and connect with others by using tools like translators. Translators are applications that automatically transcribe text or spoken words from one language to another by using algorithms and artificial intelligence (AI). This type of applications can process large amounts of data in a matter of seconds, making them ideal for providing rapid translation between two or more people. Due to the increasing popularity and usability of machine translation, this project intends to build a working AI speech translation model using state-of-the-art deep learning (DL) models and algorithms. The languages chosen for the implementation are English, Swedish, and Spanish, and the reason for their selection is them being the first languages of the project members.

The project document is structured as follows. Section 2 gives an insight into relevant related work and describes how it relates to the proposed project. Section 3 describes the methodology used and the materials selected to achieve the final implementation. Section 3 also provides a detailed outline of the execution of the project. Section 4 presents the expected results of the project and their significance in the field of embedded AI. Section 5 provides an interpretation of the results shown in the previous section, and discusses the limitations of the project as well as its achievements in its execution. Finally, section 6 summarizes the implementation of the project, revisits the methodology and results obtained, and highlights the significance and impact of the final system.

2 Related Work

This project section aims to review existing literature to contextualize and justify the proposed implementation. To understand the different aspects of the project, the review focuses on its key components: speech recognition, speech-to-text, text translation, and smart reply.

2.1 Speech Recognition

Ravanelli et al. [8] introduce an open-source, all-in-one speech-processing toolkit called Speechbrain that can be used for identifying spoken languages, and other speech-processing tasks via audio files. The toolkit features HyperPyYAML for specifying complex hyperparameters and enhancing flexibility in experiments. It can handle different lengths of sentence sequence embeddings and transformations. It supports dynamic batching and it also includes a general training loop. Speechbrain is a common template used for more specified speech-processing models, where the specified models are extensions from Speechbrain and they tend to give state-of-the-art results for speech-processing tasks.

2.2 Speech-to-Text

Machine transcription is the process of matching a generated sequence of words with a given audio signal [2]. This is achieved by using transformers, a commonly used state-of-the-art implementation. Currently, one of the best-performing models is Whisper AI, developed by Open AI [6]. This model uses an encoder-decoder transformer due to its reliable scalability [10]. Whisper AI was trained using 680,000 hours of multilingual data, which resulted in great model generalization and the ability to handle up to 57 languages.

2.3 Text Translation

Text translation is the process of converting text from one language to another while preserving the original context and meaning. In the context of neural machine translation (NMT), the process includes an encoder for source sentence representation and a decoder for generating target words. Luong et al.[5] discuss the use of various recurrent neural network (RNN) architectures, such as standard hidden units, convolutional neural networks (CNN), and long short-term memory (LSTM) or gated recurrent unit (GRU) units. The paper introduces a softmax parameterization for decoding probabilities and incorporates an attention mechanism that considers source hidden states throughout the translation. Luong et al. categorize attention models into global and local, differentiating based on whether attention is applied to all positions (global) or only a few (local). The global attention model considers all encoder hidden states, forming an alignment vector and a context vector through various content-based functions. Luong et al.[5] propose a local attention mechanism as a more computationally efficient alternative to the global attention mechanism for translating longer sequences. Inspired by the soft and hard attention trade-off in image caption generation, the local attention focuses on a small subset of source positions for each target word, making it differentiable and easier to train than hard attention.

2.4 Smart Reply

Zhang et al. [11] implement a large-scale, transformer-based neural network open-source model designed to generate conversational responses as a chatbot called DialoGPT. It is an extension of OpenAI's GPT-2 model, which was trained on 147 million Reddit comments from 2005 to 2017. DialoGPT is implemented to give relevant, contentful, and context-consistent responses from a user's input. This can be done since the model can store the chat history to make it able to understand the context of the conversation. The transformer architecture allows the model to better handle long-term contextual information in dialogues. The model can be accessed in three different states, large, medium, and small, where the large model is the biggest in size and highest in accuracy. In contrast, the small model is the most memory efficient but performs worse than the other two. By releasing three different states of the model, the model can be implemented from large cloud services to edge devices.

3 Methods and Materials

The objective of the project is to create an embedded AI speech translation system. To do so, the proposed approach consists of a five-module implementation. It combines a speech input algorithm, a speech recognition system, a

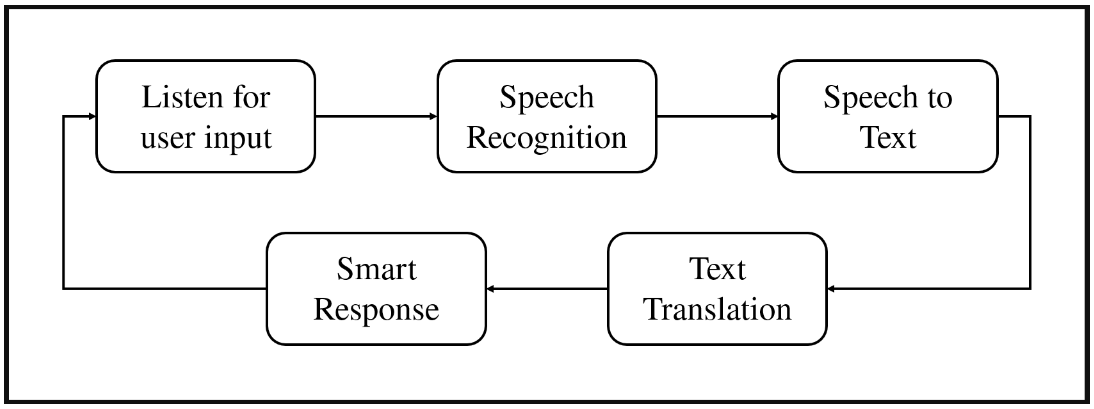


Figure 1: Speech translation system architecture.

speech-to-text model, a text translation system, and finally, an intelligent reply model capable of suggesting answers related to the context of a conversation. Figure 1 depicts the architecture of the speech translation system.

The speech-to-text transformer model was trained using an A100 Nvidia Tensor Core GPU on a compute server. While the other models were trained using a single Nvidia Geforce 3090 GPU card.

3.1 Speech Recognition

Speech recognition is the task of identifying a spoken language given an audio file. For this project, an extension of the *Speechbrain* model [8] called *lang-id-voxlingua107-ecapa* was initially chosen for classifying languages given an audio file. The model can identify 107 different spoken languages and still be able to run on an edge device. The extension has been trained on the *VoxLingua107* dataset with a cross-entropy loss function. It uses the ECAPA-TDNN architecture but it includes additional fully connected hidden layers after the embedding layer, as opposed to the original Speechbrain model. For classifying audio, the model accepts 16 kHz recordings, but if the audio file is recorded at a different frequency, it will automatically be normalized to fit the input parameters. Unfortunately, using this model resulted in incompatibility issues with the other components of this project and was replaced with OpenAI's Whisper to classify spoken languages.

3.2 Speech-To-Text

Transcribing audio using AI can be treated as a sequence-to-sequence problem and to solve this an attempt to use transformers was made. The transformer implementation was based on a guide from Keras [4]. This guide implemented a transformer model similar to the ones described in the work proposed by Vaswani et al. [10]. This guide uses an English-only dataset, therefore a different dataset was extracted from HuggingFace was used. The dataset is called CommonVoice and is from the Mozilla Foundation [1]. It contains 17,127 hours of validated and transcribed data in 104 languages. The transformer model from Keras [4] could not achieve satisfactory results within the timeframe, therefore, a fallback solution to use OpenAi's - Whisper AI [6] was used. This model is based on the paper written by Radford et al. [7]. This prebuilt model performs transcription tasks at an acceptable level while also including the built-in option to retrieve the identified language.

3.3 Text Translation

The selected model based on [9] for text translation uses a sequence-to-sequence architecture. The encoder comprises an embedding layer with a vocabulary size of 8,000 words of the source language and a GRU with 256 neurons. While the decoder, which shares the same GRU layer, implements an identical embedding architecture for the target language. In addition, a cross-attention (CA) layer and a fully connected layer are added one after the other. The CA comprises a multi-head attention layer, a normalization layer, and a concatenation layer. The MHA gets the attention output and scores, the attention scores are stored in the attention weights, and the output is concatenated to the GRU logits and normalized. Finally, the fully connected layer generated the prediction for the next token. The 4 Models were trained on the 350,000 instances of the covost2 dataset for English to Swedish, English to Spanish, Spanish to English, and Swedish to English.

3.4 Smart Reply

The model used for response suggestion in this project is the DialoGPT medium model [11]. The medium model was chosen since it has a noticeably better performance for suggesting replies than the small model, and it can still run on an edge device. Although DialoGPT has the option to store the dialogue history for better contextual understanding, it was not included in the final iteration of the product. The reason is that it requires the model to always be loaded which can lead to memory issues when the other components of this project need to be working simultaneously. The model accepts a string as an input, which can be a question or a response to a question, and returns one response as a suggestion. Even without storing the history of the conversation, DialoGPT yields acceptable outputs to fit within a conversation.

4 Results

On par with the methodology outlined in the previous section, this section of the project showcases the results obtained, including a detailed explanation of the functionality provided by every one of the system modules: the speech-to-text model, the text translation system, and the smart reply model.

The project implements a basic speech translation system that works with three different languages: English, Spanish, and Swedish. By using English as its base, it can simulate a conversation between a native speaker and another who only knows Swedish or Spanish. The system uses the three modules previously mentioned to implement the final functionality. First, it captures the user's audio input and forwards it to the speech-to-text model, which transforms the audio file into text and identifies the language. Secondly, the captured language is used to detect what the system should translate the text to, which it does by using one of the four text translation models. Finally, the model outputs a translation response that is fed into the smart reply system which generates a suggested response that can be used in the conversation. Once these steps are completed, the system loop starts again listening to a user input.

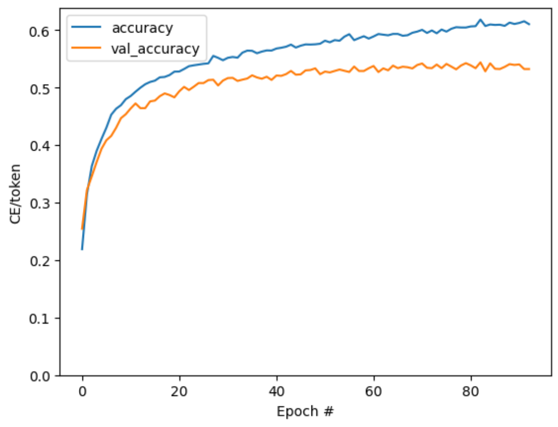
The speech-to-text model uses Whisper AI. For all specifics of how Whisper AI was built and trained we refer to the paper written by Radford et al. [7]. Whisper AI works by first converting the audio waveform into a log mel spectrogram. This is done to transform an audio file, which is in the time domain, into the frequency domain which is readable and meaningful for humans. This signal preprocessing step removes unnecessary information that would confuse any AI [3]. Once the step is completed, the log mel spectrogram is either cut or padded to be 30 seconds long. After that, the model identifies the language to retrieve the relevant vocabulary, which also serves as an important parameter for the model to transcribe correctly. When the language has been identified, the log mel spectrogram is run through transformer encoder and decoder blocks. The final result is a transcription of the original audio file. Both the transcription and the identified language from the Whisper AI are included in the final speech translation system.

The translation models were tested using 3,000 instances of the source language and target language of the covost2 dataset. The results are found in Table 1 for each of the translation models, comparing the base and TFLite models. The average accuracy saw little change from the compression, with a slight increase in performance in the English to Spanish and the Swedish to English models. Fig. 2 shows the masked accuracy of both the training and validation sets. Each model flattened at around 50% for the validation accuracy with Fig. 2b showing the Swedish to English model performing to less of an extent.

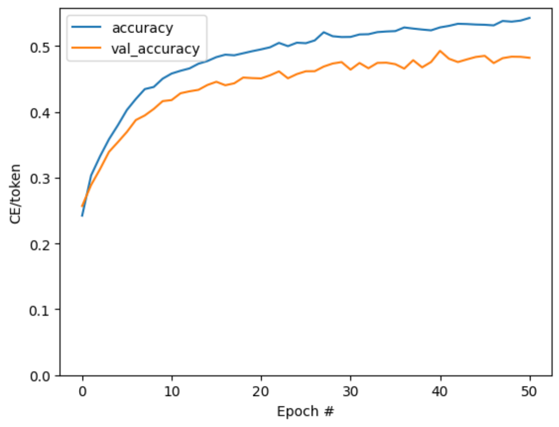
Table 1: BLEU Scores 1-Gram

Translation	Base Model	TFLite Model
English-Spanish	22.24%	22.25%
Spanish-English	21.50%	21.50%
English-Swedish	20.37%	20.37%
Swedish-English	20.92%	20.93%

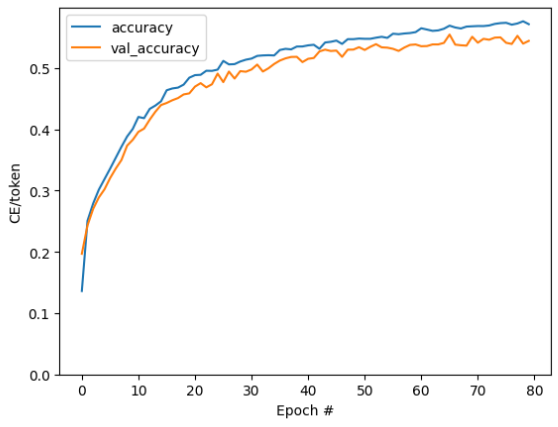
The smart reply model uses a pre-built and pre-trained model that was downloaded and stored in memory directly from the source code. The model is an open-source implementation from HuggingFace that uses PyTorch's transformers package to run inferences. A simple reply suggestion can be obtained by feeding the model a string as input. The model can store dialogue history, but since this feature was excluded from the final product, the outputs can sometimes be very generic or miss the context of the conversation. In conclusion, the model serves its purpose by generating an additional response that can be used by the users.



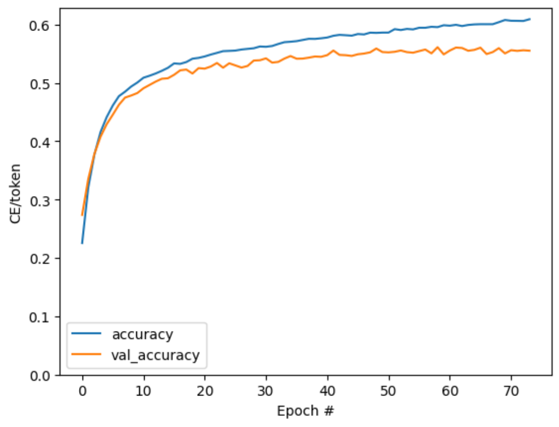
(a) English to Swedish Masked Accuracy



(b) Swedish to English Masked Accuracy



(c) English to Spanish Masked Accuracy



(d) Spanish to English Masked Accuracy

Figure 2: Masked Accuracy Translation Models

5 Discussion

The system is able to successfully implement an embedded AI program that acts as expected, translating pieces of audio into other languages. To do so, the project's methodology executes a series of modules that perform small requirements until achieving the final result. While this implementation can be considered a success, it still has much room for improvement, as the periodic execution of the different modules takes a large toll in terms of speed and memory. However, once the system loads all the different models into memory, its execution speeds up considerably. Another positive point of the system is its module-based architecture, which separates the heavy system requirements into smaller models. These separate units can be individually analyzed and improved, which facilitates future implementations by focusing on smaller independent functions at a time.

One important aspect of a project that involves several types of AI models is to make sure that every single one of them is compatible with the others. As previously mentioned, the initial speech recognition model worked and performed well on its own. However, it could not be included in the final iteration of the project due to incompatibility issues with the other modules.

The initial implementation of the speech-to-text was based on a complex transformer architecture. This version of the module was built to handle the three different languages at the same time but unfortunately encountered several problems. Firstly, the model not being capable of using one single language to process an individual audio file, which resulted in an output containing sequential words in two or more languages. This issue was overcome by building a model per language. Secondly, due to the nature of the model, it began to overfit on the validation dataset, resulting in it not being able to transcribe well enough. The proposed approach to this problem consisted of the addition of dropout layers and regularization. Moreover, new data was introduced to combat overfiting and reach better results. However, the task was so computationally expensive that it could not be completed in time, and as such, a new model proposal was built using Whisper AI.

Whisper AI is a model developed by Open AI to handle both speech-to-text and language identification [6]. This model was incorporated into the system workflow after a thorough testing of its different versions, differing in size, speed, and accuracy. Nevertheless, this implementation encountered some language identification inaccuracies, which Open AI is currently addressing [7].

The intended use of a pre-trained transformer model for text translation faced challenges in the compression process using TFLite due to unsupported operations. Consequently, a decision was made to implement the model in PyTorch. However, even after compressing the model with built-in quantization methods, it remained excessively large for deployment on an edge device. In response to these limitations, a sequence-to-sequence model was developed from scratch as an alternative to the original approach. This new model encountered challenges, particularly in achieving masked accuracy above 55% on the validation set. This limitation stemmed from the model treating unseen numbers, names, places, and words as unknown tokens, leading to difficulties in generalization. Additionally, the model struggled to retain the contextual understanding of when specific words were used for instances. In light of these observations, it is imperative to retrain the model on a more extensive corpus of translations. This training should include a broader vocabulary, and an alternative method for storing tokens must be explored to address the issues related to unseen entities and contextual understanding.

The initial plan for response suggestions involved returning four different types of responses rather than just one. These four responses had to reflect several emotional tones such as angry reply, happy reply, sarcastic reply, etc. However, smart reply models are typically aimed toward mobile platform programming languages, such as Kotlin or Swift, and require publisher-provided packages to run inference. Using these models in a Python environment requires writing custom operators in C++, otherwise inference methods can not be accessed. Implementing and training such a system was not feasible with the time constraints of this project. A different approach of prompting the DialoGPT chatbot to generate these different responses was tested to no avail. The chatbot could not process the input and generate multiple responses from it with emotional tones.

6 Conclusion

The speech translation project implements an AI-embedded system by using a module-based architecture that divides its functionality into sequential pieces of requirements. The system first obtains a user's audio input and processes it by using a speech-to-text module. This module transforms the audio file into text and obtains its language. After that, the text is forwarded to the text translation model, which transforms it into another language while maintaining its original significance. Finally, the translated text is used by a smart reply model that gives a suggested response to the aforementioned query. The system is capable of working with three different languages: English, Spanish, and Swedish.

The speech translation system is implemented by making use of several external resources such as Whisper AI and DialoGPT. However, the initial implementation consisted of self-constructed models that aimed to fulfill the requirements of the individual system components. This is achieved by the text translation models, which provide the guidelines for implementing a future system without the need to rely on other external libraries. Among other future work requirements, the project can be improved by focusing on the individual modules of the system, which are expected to provide state-of-the-art results in both speed and accuracy.

In conclusion, the project successfully implements a speech translation system able to operate in three different languages inside an embedded environment. While it may not be innovative in its respective field, it is promoting a new architecture system that can certainly be applied to multiple specific problems due to its flexibility. Moreover, the individual components of the system can be modified as development carries on, allowing for improvements that would not be possible in a monolithic architecture.

References

[1] R. Ardila et al. "Common Voice: A Massively-Multilingual Speech Corpus". In: *Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020)*. 2020, pp. 4211–4215.

[2] Santosh K Gaikwad, Bharti W Gawali, and Pravin Yannawar. "A review on speech recognition technique". In: *International Journal of Computer Applications* 10.3 (2010), pp. 16–24.

[3] Dalya Gartzman. *Getting to Know the Mel Spectrogram*. Last accessed December 21, 2023. URL: https://towardsdatascience.com/getting-to-know-the-mel-spectrogram-31bca3e2d9d0.

[4] Keras. *Keras documentation: Automatic Speech Recognition with Transformer*. Last accessed December 21, 2023. URL: https://keras.io/examples/audio/transformer_asr/.

[5] Thang Luong, Hieu Pham, and Christopher D. Manning. "Effective Approaches to Attention-based Neural Machine Translation". In: *ArXiv* abs/1508.04025 (2015). URL: https://api.semanticscholar.org/CorpusID:1998416.

[6] OpenAI. *Whisper AI*. Last accessed December 21, 2023. URL: https://github.com/openai/whisper.

[7] Alec Radford et al. "Robust speech recognition via large-scale weak supervision". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 28492–28518.

[8] Mirco Ravanelli et al. "SpeechBrain: A General-Purpose Speech Toolkit". In: (2021). arXiv:2106.04624. arXiv: 2106.04624 [eess.AS].

[9] Tensorflow. *Neural machine translation with attention*. Last accessed December 23, 2023. URL: https://www.tensorflow.org/text/tutorials/nmt_with_attention.

[10] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[11] Yizhe Zhang et al. "DialoGPT: Large-Scale Generative Pre-training for Conversational Response Generation". In: (2020). arXiv: 1911.00536 [cs.CL].