Speech Translation: Latvian-English

Hugo Albert Bonet

1 Introduction

This section provides a brief description of the selected dataset and limitations adopted to run the experiments according to our computing capabilities.

1.1 Dataset Description

The selected dataset is CoVoST 2 (Common Voice Speech Translation), a large-scale multilingual speech translation corpus covering translations from 21 languages into English and from English into 15 languages. For each language, CoVoST 2 includes different versions of audio data depending on your storing capabilities.

In our case, we selected the smaller version of the translations from Latvian (lv) to English (en). More specifically, we selected 1000 samples to perform the different experiments of speech translation (cascade and end-to-end translation).

1.2 Limitations

The experiments have been carried out using Google Colab Notebooks environment, which offers a CPU with 12.7 GiB of RAM and a GPU T4 with 15 GiB VRAM, restricted to 100 credit units of computation a month thanks to its Pro pricing plan. Therefore, some limitations have been mandatory for the proper development of the project.

Firstly, as has already been said, only 1000 audio files have been studied, from which only the ones with 275 tokens or less have been used for fine-tuning the models. For the fine-tuning process, the splits are as follows: 160 samples for training, 200 samples for validation, and 640 samples for test. Moreover, all models have been trained in the same conditions, with a batch size of 8 for 10 epochs, and a LoRA rank of 16.

2 Results

This section shows the results of the baseline and fine-tuned models on various performance metrics: Word Error Rate (WER) for automatic speech recognition (ASR), and BLEU and COMET for machine translation.

The models used for transcription are all versions of Whisper (from tiny to turbo, except base from the second experiment to the end), and an extension of a Whisper model already fine-tuned for Latvian by AiLab.lv (from now on WhisperLV). For cascade translation, different sizes of NLLB models have been used (600M, 1.3B, and 3.3B), and the model by Helsinki NLP lab as an extension. For the fine-tuning step, only Whisper tiny, Whisper large, WhisperLV, NLLB 600M, NLLB 3.3B, and Helsinki-NLP models have been combined.

2.1 Baseline Transcription

The first step is evaluating the ASR capabilities of Whisper models as a baseline. Table 1 shows the WER for each size of Whisper.

Whisper Size	WER
Tiny	98.24
Base	86.30
Small	61.32
Medium	41.96
Large	20.97
Turbo	28.03
WhisperLV	0.18

Table 1: Baseline transcription

As can be observed, models Tiny and Base are the most similar ones, apart from Turbo and Large, as Turbo is an acceleration of the Large one. WhisperLV outperforms every other model, demonstrating the power of fine-tuning.

2.2 Baseline End-to-End Translation

The Second step consists of evaluating the ST capabilities of Whisper models as a baseline. Table 2 shows BLEU and COMET for each size of Whisper.

Whisper Size	BLEU	COMET
Tiny	0.1	41.82
Base	0.1	41.90
Small	0.6	43.62
Medium	11.8	55.56
Large	17.7	63.51

Table 2: Baseline end-to-end translation

Again, Tiny and Base are practically the same. Note that models Turbo and WhisperLV do not appear, as they do not perform end-to-end translation. The maximum scores are obtained by Whisper Large.

2.3 Baseline Cascade Translation

The next step consists of evaluating the ST capabilities of a cascade architecture divided into Whisper models for ASR and NLLB and Helsinki-NLP model for MT, as a baseline. Table 3 shows BLEU and COMET for each combination.

Whisper Size	MT model	BLEU	COMET
Tiny	NLLB 600M	2.0	45.51
Small	NLLB 600M	8.0	57.39
Medium	NLLB 600M	13.0	63.71
Large	NLLB 600M	19.2	71.60
Turbo	NLLB 600M	17.2	68.26
Tiny	NLLB 1.3B	2.5	46.68
Small	NLLB 1.3B	9.6	59.17
Medium	NLLB 1.3B	15.5	66.02
Large	NLLB 1.3B	21.5	73.68
Turbo	NLLB 1.3B	20.8	70.56
Tiny	NLLB 3.3B	2.4	46.42
Small	NLLB 3.3B	9.0	58.98
Medium	NLLB 3.3B	15.2	65.75
Large	NLLB 3.3B	22.5	73.92
Turbo	NLLB 3.3B	21.0	70.61
Tiny	Helsinki-NLP	2.1	43.93
Large	Helsinki-NLP	21.8	71.46

Table 3: Baseline cascade translation

It is easily noticed that the results are better using a cascade approach than an end-to-end one, in this case, as a baseline model. Moreover, the MT model used does not appear to be significant for the results, being the majority of the variance linked to the size of the Whisper Model.

2.4 Baseline Cascade Translation

Now we fine-tune the MT models and evaluate the capabilities of a cascade system as in the previous section. Table 4 shows BLEU and COMET for each combination.

Whisper Size	MT Model	eval BLEU	test BLEU	COMET
Tiny	NLLB 600 M	3.6	2.9	50.16
Large	NLLB 600M	24.7	24.7	75.82
Tiny	NLLB 1.3B	4.3	6.6	50.95
Large	NLLB 1.3B	28.38	29.8	78.41
Tiny	NLLB 3.3B	5.2	3.8	50.31
Large	NLLB 3.3B	28.7	33.4	79.51
Large	Helsinki-NLP	26.6	20.8	66.97
WhisperLV	NLLB 3.3B	39.3	35.2	82.49

Table 4: Cascade translation with fine-tuned MT model

We can observe that the size of the MT model increases its significance once we fine-tune it, obtaining better results the bigger the model is. However, the Whisper model used continues to be the key to obtaining the best performance possible, as best fine-tunned MT model obtains significantly worse results with Whisper Tiny than the worst baseline one with Whisper Large transcription.

2.5 LV-LV fine-tuning: Error correction

Fine-tuning the MT model of the cascade system cannot only improve due to adaptation to new contexts, but also because it learns how to correct mistakes of the ASR model. Therefore, a question arises: can we train an LV-LV translation model whose task is to fix the errors of the end-to-end ST model?

Table 5 shows BLEU and COMET of the developed experiments.

Whisper Size	MT Model	eval BLEU	test BLEU	COMET
Tiny	NLLB 600 M	0.25	0.2	43.84
Large	NLLB 600M	14.4	16.4	66.66
Large	NLLB 3.3B	13.8	16.6	64.01

Table 5: End-to-end model with MT error fixer

This approach is clearly worse than the cascade system, but slightly improves the performance of the baseline end-to-end model.

3 Conclusions

Speech Translation is a complex task with multiple possible approaches. End-to-end models bring an easy-to-use solution with complete context about a task, but have to learn from scratch how to solve the whole chain of problems comprised in it. On the contrary, cascade systems provide modularity. Modules of the cascade system can be improved independently and replaced on deployment time without the need of re-training the system as an atomic model. Moreover, said modularity allows the developer to analyse the complexity of each task, focusing on the model that creates the biggest difference in performance.

However, cascade systems also have a main shortcoming: error accumulation. Mistakes are passed from model to model, which can be relevantly detrimental for the system. However, we have observed that fine-tuning modules of the system directly on the output of the previous step instead of the reference helps improve the results, teaching the module how to detect and solve previous errors.