TranslateTracks (AI Dubbing with Human-in-loop) - A Technical Report

About TranslateTracks

Given the development of highly logically capable LLMs enabling SOTA translation and human-like high fidelity AI Text-to-speech services like ElevenLabs, we could see the immense impact this would create in the dubbing industry, where both AI translation and AI Speech could completely change the industry landscape. Given this thesis in our mind, we started working on Translatetracks.

TranslateTracks ran for 7 months, did $100K in revenue with a net margin of 75%. Our Human-in-loop service stood out in the clutter of one-shot Dubbing products, where the final results are far from production ready with numerous inaccuracies, both in terms of translations and the final dubs. We created a network of translators and proofreaders for different European languages in India, thus letting us deliver the service at a fraction of the cost of our competitors. To be precise, we were 4x cheaper then our nearest competitor of comparable quality.
Here are a few videos we have dubbed:

- The Catamaran Challenge
- Tim Cook Interview (change audio track to english)
- Jamy’s Coffee Effects (change audio track to english)

Technical Description

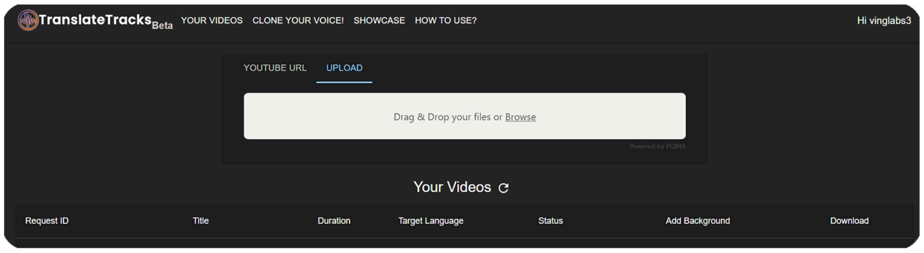
Introduction

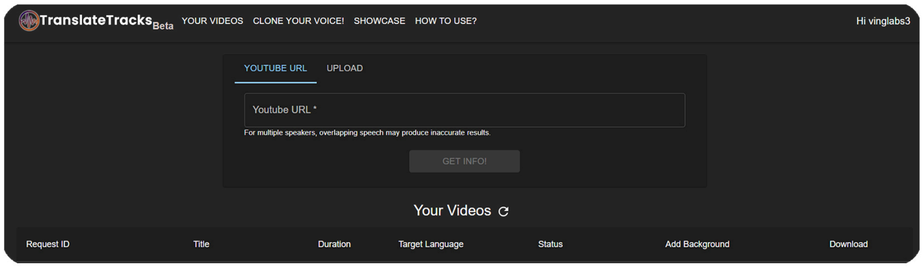
Our platform/app basically involves four different steps:

- **Video Upload** : Direct video upload/ Upload video via Youtube link. We extract the Audio from the video file. This is followed by a vocal remover process that separates the Vocal track of the audio from the Background audio, giving us a Vocal Track and a Background Track.
- **Transcription** : The separated Vocal Track is transcribed using our ASR Engine (Automatic Speech Recognition) that uses multiple SOTA transcription APIs like AssemblyAI, Deepgram and Gladia.
- **Translation** : Once the Video is transcribed, and the segments are verified, we translate the text from source language to the target language, using an LLM of our choice (Typically Chatgpt4, Claude, and LLama2). This step involves multiple LLM based optimizations, to assure that the translations are up to the mark.
- **Text-to-Speech** : This is the last step. For each translated segment, and based on the assigned speaker to the text, an AI speech is generated using a TTS API (Elevenlabs) that is matched with the characteristics of the original voice. This step also has multiple ML/AI optimization pipelines, that makes sure that final audio is expressive and engaging, and at no point, feel machine generated.

Video Upload

Our app can be accessed here. Video for dubbing can be uploaded directly or directly via a youtube link.



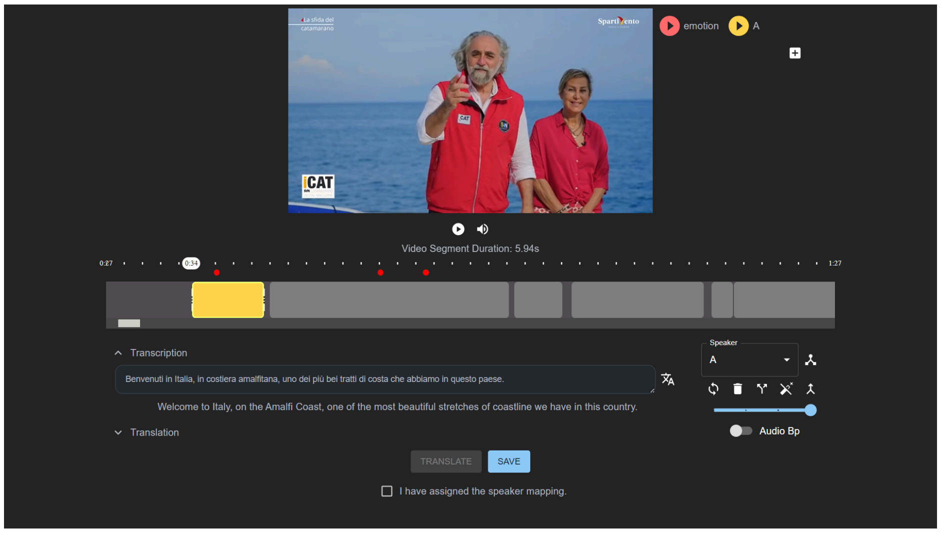


Our app is hosted on AWS EC2 Ubuntu 22 Server, backend is written in Python FastAPI,and frontend is written in React, and for the database, we are using MongoDB for storing video/user data, audios/videos are hosted on S3. We use various LLMs like ChatGPT, Claude and LLama2 in our application. All the LLM calls are logged in Langfuse for observability, LLM performance monitoring, and running evaluation on output afterwards. We are also using PostgresQL for logging user platform usage/story, and NewRelic for instance health monitoring.

Once the video is uploaded, we use a Background remover that is deployed on the same server. For every MP4 file, we get Vocal & Bg audio tracks in WAV format. This vocal file ensures that all the background music/noise is separated from the audio track. This ensures high accuracy in the Transcription process as the audio is Noise-free.

Transcription

- We then transcribe the audio file to convert it into text format. Along with generating text, this process also assigns a unique speaker label to each text segment, this process is called Speaker Diarization.
- For this we use a combination of different Speech to text APIs to get the most accurate transcripts for our video. We use Deepgram, AssemblyAI and Gladia for transcription. Every service has its own merits and demerits.
- We use our ASR Engine (Automatic speech recognition), that takes input from all three services, and optimises the output, so that we can minimise the Word Error rate, Reduce Hallucinations and maximise the Diarisation accuracy.
- All the optimization pipelines in our application results in one thing, and that is Lip Synchronized / Lip matched final Audio.
- To ensure this, we split the transcript segments into smaller sub-segments, so that each segment appropriately matches with the content on the screen.
- We automatically split these segments at Audio and Video Breakpoints. These are the points where either the Silence is greater than Threshold, or there is scene change.

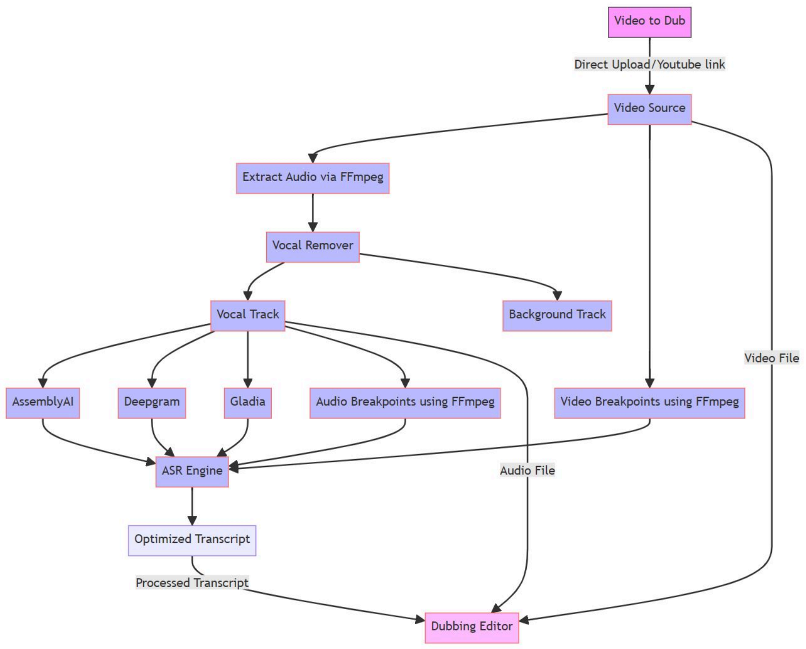


Benvenuti in Italia, in costiera amalfitana, uno dei più bei tratti di costa che abbiamo in questo paese.

Welcome to Italy, on the Amalfi Coast, one of the most beautiful stretches of coastline we have in this country.

☐ I have assigned the speaker mapping.

- Finally, we see the segments in the timeline of the editor, along with the Video and audio. Now we have a human in the loop to correct the Transcripts. The editor provides multiple functionalities to modify various aspects of the transcripts.

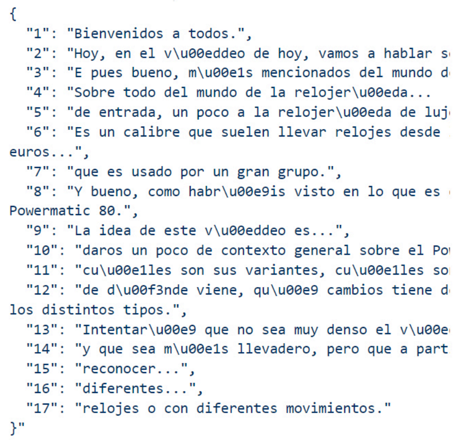


Technologies used -

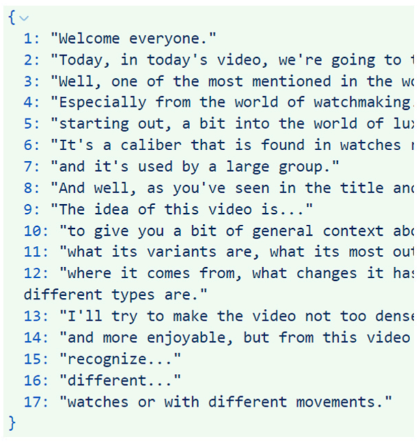
- ffmpeg - For extracting audio from video and to detect Audio and Video breakpoints.
- ASR services - Assembly AI, Deepgram, Gladia.
- VocalRemover(https://vocalremover.org/) - A service to seperate vocals from background.
- Python Libraries - moviepy,pydub

Translation

- Once transcription is verified, the next step is Translation.
- The most fundamental requirement here is that, the #segments in Transcript and #Segments in Translation should be the same.

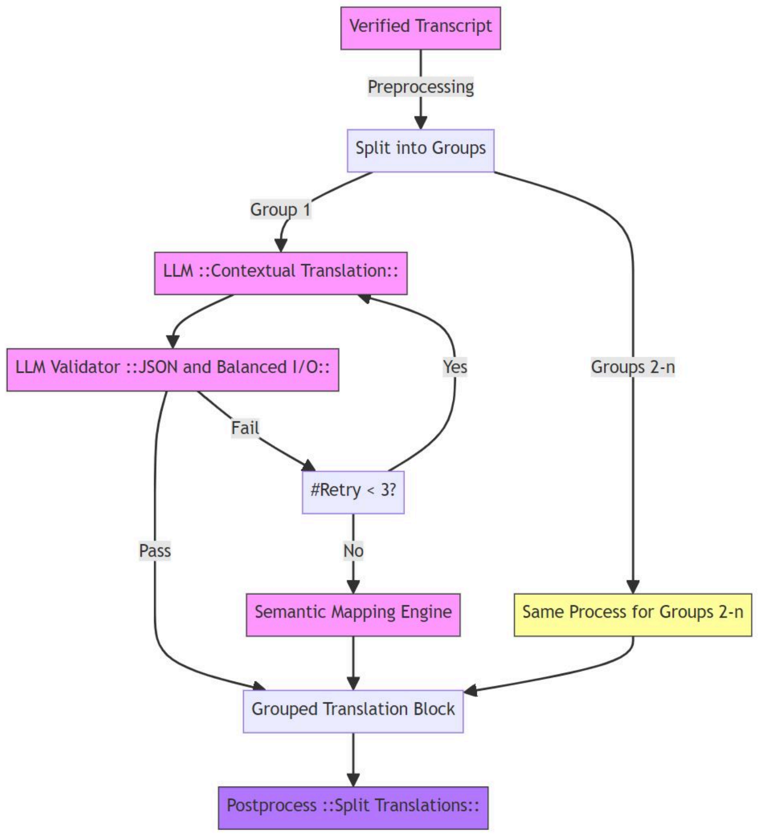


```
{
  "1": "Bienvenidos a todos.",
  "2": "Hoy, en el v\u00eddeo de hoy, vamos a hablar s
  "3": "E pues bueno, m\u00e1s mencionados del mundo d
  "4": "Sobre todo del mundo de la relojer\u00eda...
  "5": "de entrada, un poco a la relojer\u00eda de luj
  "6": "Es un calibre que suelen llevar relojes desde
euros...",
  "7": "que es usado por un gran grupo.",
  "8": "Y bueno, como habr\u00e9is visto en lo que es
Powermatic 80.",
  "9": "La idea de este v\u00eddeo es...",
  "10": "daros un poco de contexto general sobre el Po
  "11": "cu\u00e1les son sus variantes, cu\u00e1les so
  "12": "de d\u00f3nde viene, qu\u00e9 cambios tiene d
los distintos tipos.",
  "13": "Intentar\u00e9 que no sea muy denso el v\u00e
  "14": "y que sea m\u00e1s llevadero, pero que a part
  "15": "reconocer...",
  "16": "diferentes...",
  "17": "relojes o con diferentes movimientos."
}"
```



```
{
  1: "Welcome everyone."
  2: "Today, in today's video, we're going to
  3: "Well, one of the most mentioned in the w
  4: "Especially from the world of watchmaking
  5: "starting out, a bit into the world of lu
  6: "It's a caliber that is found in watches
  7: "and it's used by a large group."
  8: "And well, as you've seen in the title an
  9: "The idea of this video is..."
  10: "to give you a bit of general context ab
  11: "what its variants are, what its most ou
  12: "where it comes from, what changes it ha
  different types are."
  13: "I'll try to make the video not too dens
  14: "and more enjoyable, but from this video
  15: "recognize..."
  16: "different..."
  17: "watches or with different movements."
}
```

- We convert the input transcript to a JSON, split them into groups of maximum 50 segments, and have overlaps in segment groups to preserve translation context. For groups with more than 50 segments, the LLMs suffer from a needle in a stack issue, with poor performance, unequal input outputs, and sometimes exceed input tokens limit.


Verified Transcript
Preprocessing
Split into Groups
Group 1
LLM ::Contextual Translation::
LLM Validator ::JSON and Balanced I/O::
Yes
Groups 2-n
Fail
#Retry < 3?
Pass
No
Semantic Mapping Engine
Same Process for Groups 2-n
Grouped Translation Block
Postprocess ::Split Translations::

Technologies used -

- LLM - OpenAI gpt-4 for contextual translation
- Orchestrator - Langchain(LLMChain,FewShotPromptTemplate,Ret
- Embeddings - gpt-3.5 embeddings(small 1536 di mapping engine to map input and output segments i RetryOutputParser fails.
- Python Libraries - Langchain

- Every group is translated in one LLM prompt, so that Translation context is preserved. We orchestrate this using Langchain, with a Few Shots example, along with Output Parser to force the output to the defined formats.
- We have a separate Semantic Mapping Engine, to map the input transcript and output translation in case there is a mismatch between the number of segments in reference and translated group.
- Once each group is translated, we recombine the groups to get the translated segments. Every segment is assigned a speaker, that is mapped to an AI voice.

- Now the main challenge with translation is that the same text, when translated across languages, has different numbers of characters, and therefore different time to say the same thing. For example:

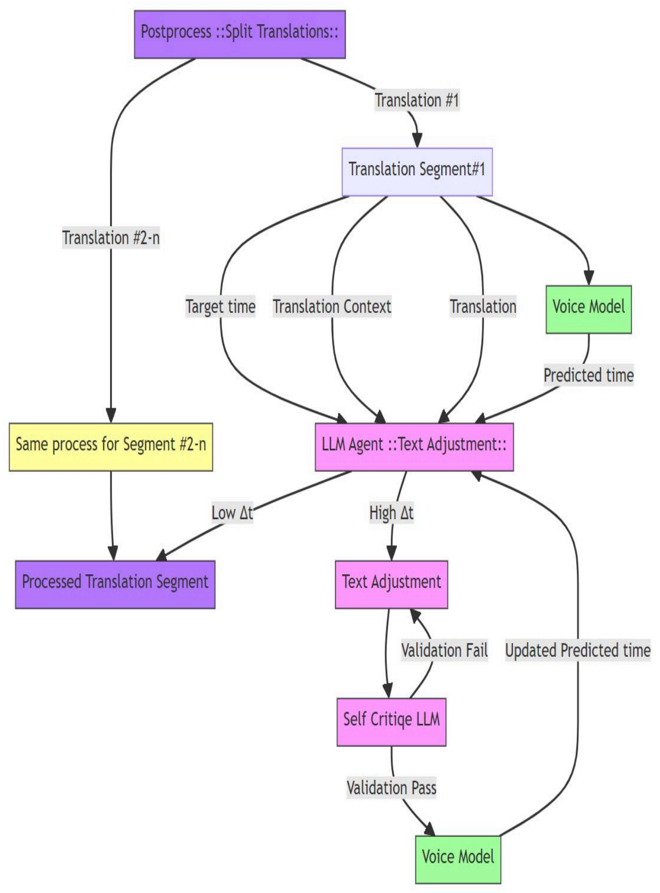
0:07 / 0:07

English

0:09 / 0:09

Hindi

- To handle this, we have a separate LLM engine that utilises our voice models. For every voice, we have developed ML models across languages to predict with ~80% accuracy, the expected duration of the generated audio from a particular text.



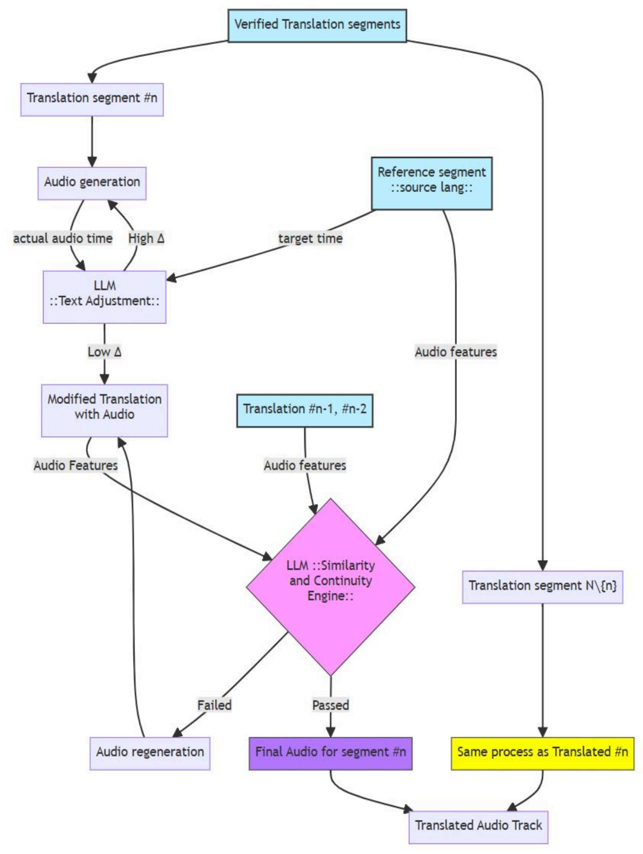
Technologies used -

- scikit-learn, pytorch - LinerRegressionModel and SVM for predicting duration of audio.(voice models)
- Orchestrator - Langchain- A custom agent(AgentExecutor) - CATTM(Context Aware Text Time Matcher) with 3 custom tools(@tools)-
  - Text Adjustment Tool - A tool built with LLMChain which takes input as contexual translation and gives out smaller or larger text as required.
  - GPT-4 evaluator Tool - A tool built with load_evaluator which uses gpt-4 to critique the quality of adjusted text given by Text Adjustment tool.
  - Voice Model Tool - A tool that takes in text and speaker as input and spits out the time required to speak that text by that speaker.
- LLM - LLama2-7b for text adjustment(finetuned using LORA), GPT-4 for evaluating outputs of llama2-7b.
- HuggingFace - PEFT for finetuning llam2-7b using LORA on custom dataset of shortening and lengthening text.

- Utilising those models, we run this process for every translated segment, and modify the translation based on the acceptable limits of Δt, for every voice.
- We have a LLM agent to adjust the translation, and a Self critique LLM to judge the modified translation on parameters such as Grammar, Context loss, Colloquialism etc. Once the translations are modified, these are reviewed by a Human in the loop for final validation.

Audio Generation

- Once the translations are verified, we generate audio for each translated segment. To ensure that the final dubbing is accurate, engaging and preserves the emotions from the original track, we have another LLM engine, that adjusts translation based on continuous feedback, and regenerates audio multiple times to achieve this target.



Technologies used -

- ElevenLabs - For Text to Speech and voice cloning.
- Orchestrator - Langchain - CATTM agent
- Wav2vec and SpeechBrain - For Audio similarity and continuity engine.
- Python Libraries - Langchain, SpeechBrain, pydub, moviepy

- In this process, we have a Similarity and continuity engine that takes audio features from previous generated segments, as well as features from the reference audio, and regenerates audio multiple times to ensure that the output audio has similar emotions to the source segment, and when played with previous segments, it should sound continuous.
- Once this process is done for every segment, the generated audios are verified by our Human experts, and they ensure the final Dubbing is perfect.