BABEŞ-BOLYAI UNIVERSITY CLUJ-NAPOCA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
SPECIALIZATION COMPUTER SCIENCE IN ENGLISH

DIPLOMA THESIS

Enhancing Romanian Speech Recognition by Using Cross-Lingual Data from Romance Languages

Supervisor
Assist. PhD. Briciu Anamaria

Author
Selegean Victor

2025

UNIVERSITATEA BABEŞ-BOLYAI CLUJ-NAPOCA
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ
SPECIALIZAREA INFORMATICĂ ÎN LIMBA ENGLEZĂ

LUCRARE DE LICENŢĂ

Îmbunătățirea Recunoașterii Vocale pentru Limba Română Folosind Date Interlingvistice Provenite de la Limbi Latine

Conducător științific
Assist. Dr. Briciu Anamaria

Absolvent
Selegean Victor

2025

ABSTRACT

Automatic Speech Recognition is a difficult task, requiring vast amounts of training data and compute time in order to produce proper results. This is especially difficult for languages with moderate numbers of speakers, for which current tooling and resources seem to lag behind.

This paper documents an investigation into the effects of incorporating speech data from multiple romance languages as a means of improving the accuracy of Romanian Automatic Speech Recognition (ASR) systems.

Previous studies have shown that augmenting the data used in training such a system gives better results when working within the same language family. In particular, it was showed that such an enhancement can be made for Romanian using Spanish resources. A similar result was achieved with Italian. However, it was also also noticed that the performance degrades once the cross-lingual data increases in percentage over a certain threshold.

Building upon these findings, this paper will explore the usage of speech data from multiple related languages simultaneously. It will explore whether such an augmentation of the Romanian ASR training datasets can further enhance model accuracy. In addition, the investigation will attempt to answer whether the inclusion of multiple languages allows the training set to accommodate a higher percentage of cross-lingual data before seeing the performance degrade.

Contents

Chapter 1

Introduction

The task of Automatic Speech Recognition, abbreviated **ASR**, so prevalent today, in the age of personal assistants, is highly dependent on the language that is interpreted and on the quality and quantity of said language's resources. Languages such as English, Mandarin, or Spanish benefit from hundreds of millions of speakers and the vast amounts of data generated by them. As a direct result, the systems specialized in these languages are much more accurate than their counterparts for languages with modest speaker counts, like Romanian. This paper attempts to bridge this gap by using data and technologies afferent to Spanish and Italian in order to improve the performance of Romanian systems. If this proves successful, it may provide a framework for developing further technologies and tooling for other languages with limited resources.

1.1 Research Questions

Across this paper, the following questions will be answered:

Question 1: How does the incorporation of multiple different languages as a basis for Romanian ASR affect the final system's performance?

Question 2: If the performance of the ASR systems can be improved, is there a limit to how much Spanish and Italian data we can introduce before the performance starts to degrade?

Question 3: If such a limit exists, is there an ideal ratio that maximizes the system's performance?

Question 4: How do differing degrees of Italian and Spanish interference in the Romanian ASR systems perform in relation to each other?

1.2 Original Contribution

Experimenting with a small multilingual basis for a Romanian speech recognition system represents a novel approach in a landscape dominated by massive models and heavy computational costs.

The ecosystem of tools for Romanian language processing is still in its infancy compared to the more mature ecosystems available for English, Spanish, or Italian. During the course of this investigation, a suite of new tools and resources was developed in order to further advance the Romanian language processing capabilities.

A model for converting phonetic transcriptions into Romanian text was created and is available as part of the **Epigraph**[1] suite of tools and scripts. A pipeline for transforming Italian and Spanish text into its equivalent Romanian phonetic transcription was developed as part of the same **Epigraph** [2] suite.

Using the Epigraph suite of tools, a collection 12 of small datasets was created for the purpose of fine-tuning Romanian ASR models. These datasets were published on Hugging Face Hub on the profile **victors3136**[3]. For each of the aforementioned datasets, a small model was fine-tuned and published on Hugging Face Hub, also under the same **victors3136**[4] profile.

One of the models was deployed on **Inference Endpoints**[5], a platform offering Functions-as-a-Service. For the purpose of interfacing with the inference endpoint, a server, **Epigraph Online**[6], was created and hosted on AWS. Tied to the server, an S3 bucket was set up to store audio files of consenting users requesting a transcription. In time, this bucket can serve as a data source for new and improved Romanian ASR systems.

For simplifying server interactions, an Android application, **Epigraph Mobile**[7], was created, allowing users to record audio, request transcriptions, and consent to store their recordings directly from their mobile phone. The application can be downloaded from the website Epigraph Online.

[1]https://github.com/victors3136/Epigraph-Model/blob/main/Processor/DeepGraphemizer/phoneme2grapheme_converter.py

[2]https://github.com/victors3136/Epigraph-Model/blob/main/Processor/pipeline.py

[3]https://huggingface.co/datasets?&search=victors3136

[4]https://huggingface.co/models?&search=victors3136

[5]https://endpoints.huggingface.co/victors3136/endpoints/whisper-model-small-ro-finet-ryi

[6]http://ec2-13-60-99-27.eu-north-1.compute.amazonaws.com:8000/

[7]https://github.com/victors3136/Epigraph-Mobile

1.3 Artificial Intelligence Tools' Usage Disclaimer

During the preparation of this thesis, the author used ChatGPT in order to assist with proofreading, grammar correction, and formulation feedback. Additionally, DeepSeek and ChatGPT were used in order to identify and resolve issues related to Docker and AWS during the application development phase. After using these tools, the author reviewed and edited the content as needed and takes full responsibility for the content of the thesis.

1.4 Structure of the Paper

The rest of this paper is structured in 4 parts.

Chapter 2 represents an overview of the history and main concepts that will be relevant over the rest of the study. It offers a brief overview of automated speech recognition and phonetics before reviewing several previous studies and comparing available datasets for the task of Romanian speech recognition.

Chapter 3 contains the setup, experiment and results. It chronicles the tool and dataset choices made over the course of working on this project. It describes the building of several models and the way in which their performances were measured.

Chapter 4 presents the application. It gives details on the deployment of one of the models developed in the previous chapter. It also describes how the files sent to the model can be stored and used to create better datasets for Romanian speech recognition systems.

Chapter 5 highlights the merits and achievements of this paper. It analyses some of the faults and areas in which the presented stratedy can be further improved.

Chapter 2

Theoretical Background

2.1 Theoretical Basis

For the purpose of clarity, the following section will describe the terminology used.

2.1.1 Automatic Speech Recognition

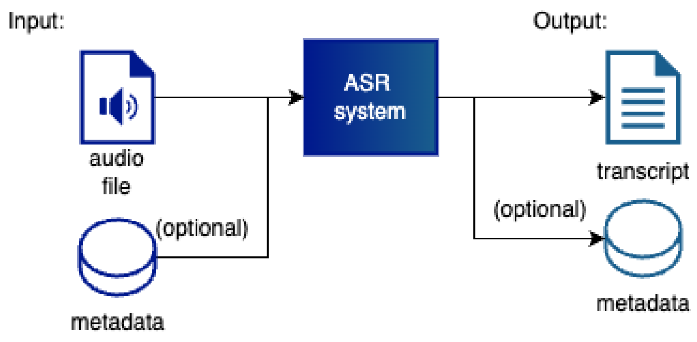


Figure 2.1: Formal definition of ASR

Figure 2.1 represents a generic definition of the ASR process. At its core, the process handles the transformation of spoken audio into written text. As such, it is also referred to as **speech-2-text conversion** or **machine transcription**. The input data may also contain information about the speaker's age, gender, or, in the context of multilingual models, the language spoken. On the output side, the system may produce metadata as well, with details such as timestamps (useful in automatic caption generation).

2.1.2 Classification of Languages

Over the course of this paper, certain languages will be categorized as **high resource** and others as **low resource** in relation to the amount of data available for said language. It is important to note that both of these terms are inherently relative. Some languages like English or Spanish have an overwhelming quantity of data, making them universally regarded as high resource. The concept becomes more subjective for languages like Italian and Romanian. Italian can be categorized as low resource when compared to Spanish, but as high resource when compared to Romanian. In turn, Romanian may be low resource when compared to Italian, but high resource when compared to Aromanian.

2.1.3 Transfer Learning in the context of ASR

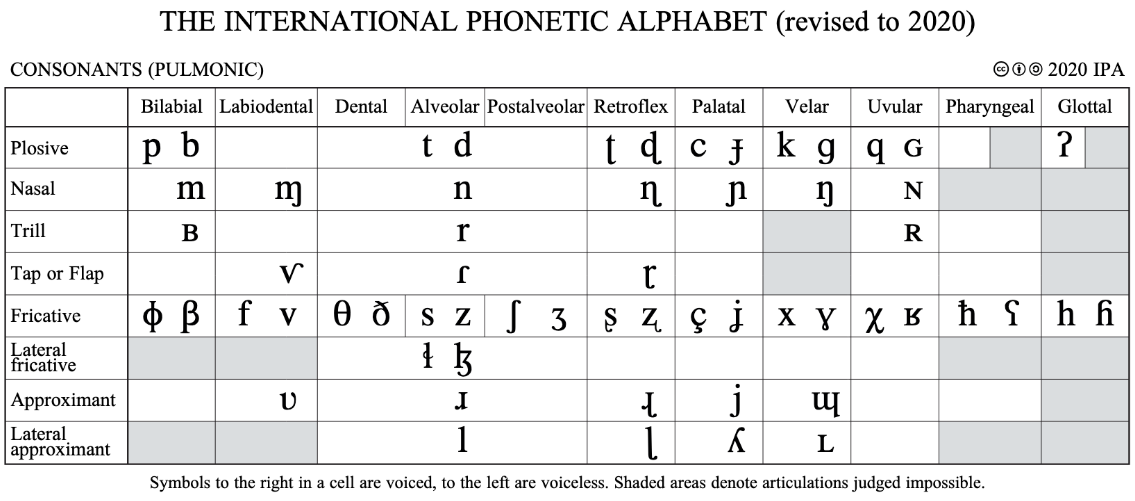
Transfer learning refers to applying the knowledge gained from solving one task to solving a different, but related, one. For the purpose at hand, the initial task is transcribing a high resource language. Later on, the initial language will be referred to as the **source language**. The related task will be the transcribing of a different language. That language will be referred to as the **target language**.

2.1.4 Linguistic Concepts

Within the broader field of linguistics, several key concepts will prove useful in the following chapters.

A **grapheme** refers to the smallest functional unit of a writing system, in short, a character. A **phoneme** refers to a set of similar speech sounds that are perceived by a speaker of a language as a single sound. Very few languages have a 1-to-1 mapping between their phonemes and graphemes. In Romanian, the same phoneme, /c/, may be written with the grapheme "c", as in "**ch**in" or with the grapheme "k" as in "**k**ilogram".

The International Phonetic Alphabet [Ass] (**IPA**) is an alphabet created by the International Phonetic Association. Its purpose is to ascribe a symbol for each phoneme. Thus, it is possible to create an objective transcription of an oral sound without relying on the rules and pronunciation of any other language. In order to clearly distinguish between normal text and phonetic transcriptions, the convention is that phonetic transcriptions are written between forward slashes("/"). The sounds most common in European languages, such as /s/ and /t/ are represented by the same characters in the IPA. Other, less common sounds such as the one the grapheme group "th" makes in the English word "**th**ing" have symbols borrowed from Greek, /θ/. A symbol that will be used later in this paper is /ʃ/, representing the Romanian "ș" sound, or the English "sh". Together with a /t/, it produces the sound /tʃ/, which represents the sound made by the Romanian "ce" or English "ch" (in "**ce**ară" and "**ch**ain" respectively). There are many more rules and symbols, but they are beyond the scope of this paper. Figure 2.2 presents the chart of all pulmonic (made by pushing air from the lungs) consonants recognized by the International Phonetic Association.



THE INTERNATIONAL PHONETIC ALPHABET (revised to 2020)

CONSONANTS (PULMONIC) ©①◎ 2020 IPA

	Bilabial	Labiodental	Dental	Alveolar	Postalveolar	Retroflex	Palatal	Velar	Uvular	Pharyngeal	Glottal
Plosive	p b			t d		ʈ ɖ	c ɟ	k ɡ	q ɢ	▨	ʔ ▨
Nasal	m	ɱ		n		ɳ	ɲ	ŋ	ɴ	▨	▨
Trill	ʙ			r				▨	ʀ		▨
Tap or Flap		ⱱ		ɾ		ɽ		▨			▨
Fricative	ɸ β	f v	θ ð	s z	ʃ ʒ	ʂ ʐ	ç ʝ	x ɣ	χ ʁ	ħ ʕ	h ɦ
Lateral fricative	▨	▨		ɬ ɮ						▨	▨
Approximant		ʋ		ɹ		ɻ	j	ɰ			▨
Lateral approximant	▨	▨		l		ɭ	ʎ	ʟ		▨	▨

Symbols to the right in a cell are voiced, to the left are voiceless. Shaded areas denote articulations judged impossible.

Figure 2.2: IPA Pulmonic Consonants Chart

The process of grapheme to phoneme conversion, abbreviated **G2P**, involves transcribing text from a given language using the IPA into language-agnostic phoneme sequences. The inverse process, of phoneme to grapheme conversion, **P2G**, is the inverse of that: provided a sequence of phonemes, produces a language-specific transcription.

To clarify these concepts, one may consider the English word "chart". Passed through a G2P conversion, it may produce the phoneme sequence /tʃart/. When that is further passed through a Romanian P2G conversion, it may produce something similar to "ceart". Note that "ceart" is not a real Romanian word, but corresponds to how a native Romanian speaker would hear and interpret the sounds made by an English speaker pronouncing "chart".

2.2 Literature Review

2.2.1 Overview of Speech Recognition Systems

Speech recognition has been an important part of natural language processing since the early days of computer science and artificial intelligence as disciplines of study, efforts dating back to the early 1950s at Bell Labs. The past decades in particular saw great strides made in the domain, owing mainly to advances in deep learning and the proliferation of large-scale datasets. 2008 saw the release of CALO[TSV$^+$08], the 'Cognitive Assistant that Learns and Organizes', a system attempting to integrate multiple AI technologies into a virtual assistant. The project paved the way for the Siri assistant used by Apple within their products starting in 2011. Despite these advances, progress was made predominantly in the context of high-resource languages. A large numbers of speakers, extensive research backing, large datasets, and an abundance of both diverse and historical data can make a language like Spanish a prime candidate for performant ASR systems.

Cross Language Speech Recognition, often abbreviated as **XLS-R**, represents a set of technologies that aim to use data and tools specialized for these high-resource languages to improve the performance of tools for languages with lower resources. XLS-R must deal with a number of specific challenges. One of the key challenges is data scarcity: a lack of extensive or properly-labeled datasets hinders traditional supervised learning techniques. Even further, phonetic diversity, different "casts" of phonemes and rules by which they arrange themselves, can cause models to learn rules which do not properly transfer between languages. Not least, domain mismatch refers to the fact that models trained on domains that are very specific struggle to adapt and handle new contexts.

2.2.2 Multilingual Representation Learning

Conneau (2021)[CBC+21] introduced XLSR, a framework for learning cross-lingual speech representations through unsupervised learning using raw audio from multiple languages. Building upon wav2vec [SBCA19], the pretrained model released in conjunction with the paper (XLSR-53) showed again that low resource languages draw benefits when sharing a model with multiple resource-rich languages.

As part of the evaluation process, the study treats Italian as a low resource language by limiting the amount of Italian training data to only 5 hours. Following this baseline, 7 models were trained for another 50 hours with unlabeled data from Italian, Spanish, German, English, Russian, Kabyle, and Chinese, respectively. At the end, another hour of fine tuning on Italian labeled data was performed. The results revealed that the models trained for the extra 50 hours performed better than the model trained for only 5 hours, regardless of the language used. However, they also suggested that language similarity influences the usefulness of training on data from a certain source language. The model which used Spanish data performed 25% better than the one trained on Chinese.

2.2.3 XLSR between related languages

In 2019, Zgank [Zga19] showed that using a source language related to the target one greatly improves the performance of the final result. He showed that the same algorithm, when starting from Spanish and targeting Slovenian, had an accuracy as low as 18%, while targeting Romanian resulted in an accuracy of around 70%. In another study, Gasan and Păiș (2023) [GP23] explored the use of a source language that is even more closely related to the targeted Romanian. Specifically, they chose Italian and proposed the architecture illustrated in Figure 2.3 to convert Italian (input: audio, target: transcription) pairs into a representation that more closely resembles authentic Romanian training data.

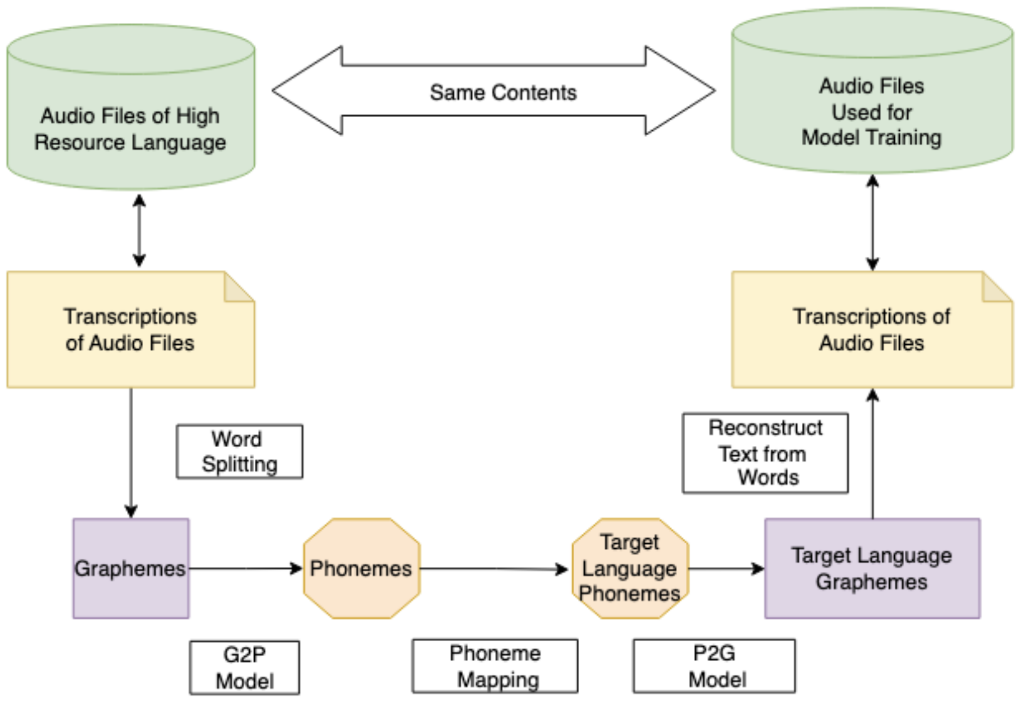

Audio Files of High Resource Language
Same Contents
Audio Files Used for Model Training
Transcriptions of Audio Files
Transcriptions of Audio Files
Word Splitting
Reconstruct Text from Words
Graphemes
Phonemes
Target Language Phonemes
Target Language Graphemes
G2P Model
Phoneme Mapping
P2G Model

Figure 2.3: Architecture Proposed by Gasan and Păiș [GP23]

The advantage of this architecture is that it assumes very little about the source and target languages. That is, as long as the appropriate G2P and P2G models exist and a mapping from the source language phonemes to the target one is put in place, it can be used for any desired language pair. This extensibility is particularly valuable in our work, as will be discussed in later sections.

2.2.4 Romanian ASR

In 2020 **RACAI**, the Research Institute for Artificial Intelligence "Mihai Drăgănescu", part of the Romanian Academy,[APT20] developed RobinASR [1], a high performance,

[1] `https://github.com/racai-ai/RobinASR`

end-to-end, speech recognition system trained on massive amounts of data (230 hours). The model achieved a word error rate [2] of 9.91% and character error rate of 2.81%, all while keeping latency on a GPU at 70ms.

The resulting model, while performant and accurate, required a very large initial time investment. With an epoch count of 70, the training process was computationally expensive and time consuming. This raises an important consideration: in a scenario where quick, basic transcriptions are needed, such a heavy model may prove excessive. Additionally, fine tuning costs grow as well. In such cases, smaller and more adaptable models may prove more practical.

2.3 Romanian ASR Data Sources

2.3.1 CoRoLa

CoRoLa (Corpus of Contemporary Romanian Language)[TBMI+19] is a large-scale reference corpus developed by the Romanian Academy's Research Institute for Artificial Intelligence. It comprises more than 103 hours of audio recordings. The corpus gathers its data from numerous and diverse sources, including texts from books, academic articles, blogs, legal documents, and oral recordings, covering over 70 subdomains.

The oral data includes studio-quality and radio speech, though spontaneous conversational speech is not present. Each entry is rich in metadata. However, the corpus itself needs to be queried using a tool such as KorAP [RACa], NLP-CQP [RACb], or OCQP [RACc].

2.3.2 Common Voice

Mozilla's **Common Voice** (CV) [ABD+20] is a multilingual, crowd-sourced speech corpus designed for open access ASR training. The Romanian segment of Common Voice contains a modest volume of validated speech (around 9 hours at the time of reporting), collected from volunteers.

However, this dataset is affected by demographic imbalances. For example, more than 80% of the samples are from male speakers, and there is very little representation of female voices over 30. Due to this fact, a model trained on this dataset may have worse results for these underrepresented peoples. Additionally, the speech is mostly read text from Wikipedia-style prompts and may not reflect natural speaking patterns.

[2]Word and character error rates will be discussed in 3.4.1 and 3.4.2, respectively

This dataset is publicly available on Hugging Face Hub[3], which makes it a good fit for quick application development.

2.3.3 USPDATRO

The USPDATRO[PBMI+24] dataset was developed to address demographic gaps found in corpora like Common Voice. It contains more than 4 hours of manually transcribed Romanian speech, collected from open multimedia sources such as YouTube, SoundCloud, and Vimeo. It specifically targets under represented voice categories, including children, women, and elderly speakers.

The recordings are predominantly spontaneous, with good audio quality. The dataset includes rich metadata (speaker age, gender, speech type), and transcriptions are manually segmented and annotated. ASR system evaluations on this corpus show significantly higher error rates compared to less balanced corpora, suggesting that there is still a gap to be filled when it comes creating tools that take into account the full range of Romanian language expression.

2.3.4 Comparison

The three corpora serve different but complementary purposes. CoRoLa is by far the most extensive and versatile, offering high-quality, well-annotated written and spoken data suitable for training large-scale models and linguistic analysis. Common Voice provides accessible, community-driven data but suffers from demographic skew and limited spontaneity. USPDATRO fills this gap by offering high-quality, spontaneous speech from under-represented groups, though on a smaller scale.

[3]`https://huggingface.co/datasets/mozilla-foundation/common_voice_11_0`

Feature	**CoRoLa**	**Romanian CV**	**USPDATRO**
Source Type	Institutional, curated	Crowd-sourced volunteer speech	Open-license multimedia
Scale (hours)	103+	9	4
Demographic Coverage	Limited info for oral data	Mostly young males	Focus on under-represented voices
Speech Type	Formal speech, studio recordings	Short read prompts	Spontaneous speech
Additional Considerations	Requires using a special domain-specific language for querying	Easily accessible via Hugging Face Hub	Needs to be manually downloaded and mapped to a dataset structure

Table 2.1: Comparison of Romanian language corpora

In summary, CoRoLa is ideal for comprehensive language modeling and analysis. Common Voice is easy to access and continually expanding but demographically limited. USPDATRO is small but crucial for fairness and representation in ASR systems. An ideal approach would be to combine these different sources in order to exploit the advantages of each and mitigate their shortcomings.

Chapter 3

ASR with Cross Lingual Data

3.1 Problem Definition

The following chapter presents an experiment aiming to answer the questions outlined in Chapter 1. To this end, cross linguistic transfer learning will be employed, with Italian and Spanish as source languages, in conjunction with some Romanian as a target language. They will be used to build datasets with differing source language configurations. Those datasets will then be used to train speech recognition models. Each model's performance will be evaluated and analyzed based on their training set's configurations. By the end, the results of the models' evaluation will be compared to asses the impact of source language choice on the resulting ASR model's performance.

3.2 Constructing a Multilingual ASR Dataset

Figure 3.1 shows an overview of the dataset generation pipeline. It extends the architecture from Figure 2.3 by adding multiple sources uniting after the phoneme mapping phase. By keeping each layer of the pipeline modular, this architecture also allows for other languages to be added. If, at a future moment, it was decided to also insert samples from Russian, all that would be required would be a Russian dataset, and the 3 language-specific steps: tokenizer, G2P conversion and phoneme mapping. The same separation of concerns would allow the swapping of Romanian as a target language by updating the phoneme mappers to the new target and providing appropriate P2G converters and reconstructors.

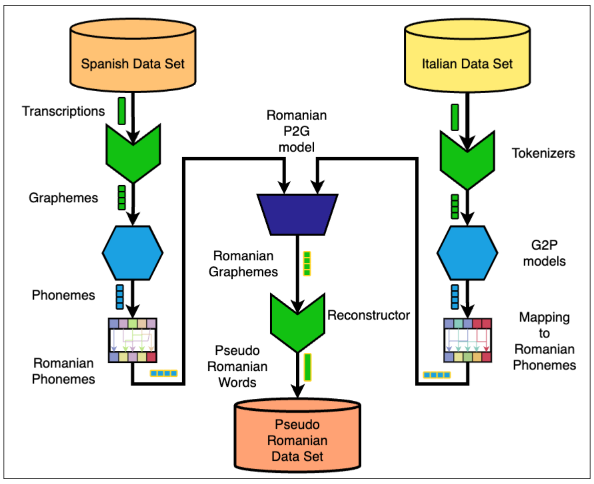


Figure 3.1: Pipeline for generating Pseudo Romanian data

By the end of the pipeline, audio recordings of Italian and Spanish speakers have Romanian-style phonetic transcriptions of text. This would allow, in theory, a model to learn to distinguish phonemes, and so, perform better at Romanian ASR.

3.2.1 Data Sources

In order to create a consistent and comprehensive strategy for our pipeline, it is useful to build upon a uniform source for foreign data. Consequently, the Common Voice [ABD+20] datasets for the high-resource base languages will be used. Version 11.0[1] of the dataset, visualized in table 3.2.1, contains over 400 validated hours of Spanish speech and 300 of Italian. Common Voice 11 also provides 17 hours of Romanian audio, firmly grounding the classification of Romanian as a low-resource language for this experiment.

Language	Recorded Hours	Validated Hours
Italian	353	327
Spanish	2077	413
Romanian	40	17

Table 3.1: CommonVoice 11.0 dataset sizes

3.2.2 Preprocessing

Similar to the work of [GP23], we must address 3 distinct preprocessing steps: word splitting and reconstruction, G2P and P2G conversions and mapping between different languages' phoneme set. At each step, special attention will be paid to uniformity across different languages.

[1] `https://commonvoice.mozilla.org/en/datasets`

Word Splitting and Reconstruction

Before applying the G2P and P2G models, all textual data is normalized and tokenized at the word level. This ensures consistent phoneme alignment and prevents errors caused by punctuation, contractions, or irregular spacing.

For Romanian, care needs to be exercised in order to preserve diacritic marks during tokenization, as they are phonemically significant and influence both pronunciation and meaning. For Spanish and Italian, accented characters (e.g., á, é, ì) will be converted to their base forms where required by the phoneme dictionary or preserved if supported natively by the model.

All languages used in this project apply whitespace-based tokenization, followed by the removal of punctuation and lower-casing, in accordance with best practices for training sequence-to-sequence G2P models.

In the final step of the pipeline, whitespaces and punctuations are restored, while applying small punctuation improvements aiming to make the final transcriptions match Romanian punctuation rules. Removing Spanish-specific punctuation, "¿" and "¡", are part of this step, as is replacing the apostrophe ("’"), the Italian contraction marker, with the Romanian hyphen ("-") when found between 2 words.

Tokenization is achieved through the use of spaCy [2]. The product of this tokenization is a spaCy Doc object, which is then used in the reconstruction step to restore whitespaces and punctuation.

G2P and P2G Conversion

Gasan and Păiș [GP23] make use of CMU Sphinx's G2P sequence-to-sequence model [CMU]. However, as of 2025 the model is unmaintained. Instead, the experiment relies on the DeepPhonemizer [spr] library for G2P conversion. This model offered a convenient API for fine-tuning with data scraped from Wiktionary for Castillian Spanish and Italian using the Wikipron [CC] library. Additional processing was required for lower-casing all graphemes and formatting the phonetic transcriptions.

For Romanian P2G conversion, a custom model was trained using data scrapped with the same library. On its basis, this is a simple text-to-text model, but in practice it is able to generalize for a language with simple spelling rules, as is the case for Romanian. Data used in training these models is available in the GitHub repository [Wik].

[2] `https://spacy.io/`

Mappings

A key step in the preprocessing pipeline is the phoneme mapping between base languages (Italian or Spanish) and Romanian. This task is necessary to ensure that the output phonemes from a G2P model in one language can be meaningfully reinterpreted and reconstructed as valid Romanian words using the P2G model.

A custom mapping table was created by aligning the IPA inventories of each base language to Romanian. For example, the Italian phoneme /tʃ/ is directly mapped to its Romanian counterpart, while others such as the palatal nasal /ɲ/ ("gn" in words like "**gn**occhi") require approximations based on contextual usage.

Where perfect 1-to-1 mappings are not available, approximate matches will suffice, chosen based on acoustic similarity and phonological function. This approach, while having certain limitations, aims to preserve the general prosody and phonemic structure necessary for training effective ASR models.

It is important to note that Italian varies greatly depending on the speaker's native dialect. What is known as literary Italian is based on the dialect spoken around Florence. A speaker from rural Naples or Sicily may not be able to comprehend a speaker of the Milanese or Venetian dialects. Spanish must also be considered, from European Spanish to the dialects spoken in California, Chile, the Philippines, and everywhere in between. For compiling the mappings, information relating to Romanian phonetics was based on a website maintained by the Technical University of Iași [Tec]. For Italian, an article from the Oxford Faculty of Linguistics, Philology and Phonology[Pao16] was used. Lastly, information about Castillian Spanish Phonology is based on an article from "Revista de Lingüística y Lenguas Aplicadas" of Rioja University [Sal10]. In the subsequent tables, a lack of notes for a row signifies that no mapping is required and the phonemes coincide between the languages.

Italian IPA	**Romanian IPA**	**Notes**
a	a	
e	e	
ɛ	e	Merged into Romanian e
i	i	
o	o	
ɔ	o	Merged into Romanian o
u	u	

Table 3.2: Italian vowels mapped to Romanian ones

Table 3.2 shows that the Italian vowel space is more diverse than the Romanian one, with the vowels /ɛ/ (as in "bene") and /ɔ/ (as in "però"). However, certain southern dialects (such as the Palermo dialect) drop the distinction between them and the familiar /e/ and /o/, respectively, entirely. The same merging is assumed to occur for a native Romanian speaker.

Spanish IPA	Romanian IPA	Notes
a	a	
e	e	
i	i	
o	o	
u	u	

Table 3.3: Spanish vowels mapped to Romanian ones

Table 3.3 highlights that Spanish and Romanian share a very similar vowel space.

Italian IPA	Romanian IPA	Notes
b	b	
d	d	
t	t	
k	k	
ɡ	ɡ	
p	p	
f	f	
v	v	
s	s	
z	z	
m	m	
n	n	
ɲ	nj	As in "Sali**gn**y" or in "bi**n**e" in some dialects
ʎ	lj	As in "**li**ană"
r	r	Rolled "r", preserved
ɾ	r	Merged into r
ʃ	ʃ	As in "**ș**arpe"
tʃ	tʃ	As in "**ce**a"
dʒ	dʒ	As in "**ge**am"
ts	ts	
dz	dz	
w	u̯	As in "**o**aie" or "sit**u**ație"
j	j	As in "**i**arăși"

Table 3.4: Italian consonants mapped to Romanian ones

Table 3.4 shows that the Italian consonant inventory has a number of symbols that are not present in standard Romanian. However, they can be closely approximated with palatalized consonants: /lj/ for the sound represented with "gl" in words like "a**gl**io", /nj/ for the sound represented with "gn" in words like "lasa**gn**a". When paired with a competent Romanian P2G model, the phoneme pair /nj/, already present in Romanian in words like "Pașca**ni**" will get accurately mapped to "ni", transforming the Italian "lasa**gn**a" into Romanian-like "laza**ni**a".

Spanish IPA	**Romanian IPA**	**Notes**
b	b	
d	d	
t	t	
k	k	
ɡ	ɡ	
p	p	
f	f	
s	s	
z	z	
m	m	
n	n	
ɲ	nj	As in "Sali**gn**y" or in "bi**ne**" in some dialects
ʎ	lj	As in "**li**ană"
r	r	Rolled "r", preserved
ɾ	r	Merged into r
tʃ	tʃ	As in "**ce**a"
x	h	Merged into h, close approximation
θ	s	Merged into "s", but "z" or "t" may fit as well
ʃ	ʃ	As in "**ș**arpe"
ʝ	j	Approximated with palatal glide

Table 3.5: Spanish consonants mapped to Romanian ones

Table 3.5 shows that, in addition to the consonants present in Italian, Spanish also has the consonants /x/ (as in "**j**uego") and /θ/ (as in "**ce**rve**z**a"). Although x has a simple and intuitive Romanian counterpart in the sound /h/, /θ/ is more difficult to accurately translate. Depending on the speaker, /θ/ may sound like /s/, /z/ , or /t/. Ultimately, the decision was taken to map it to /s/, as it matches the original in vocalization and manner of articulation: /θ/, /s/ and /z/ are pronounced without obstructing the vocal tract, by allowing airflow to pass through a narrow channel. /θ/, /s/, and /t/ are voiceless, meaning that there is no vibration of the vocal chords while articulating the sound.

3.3 Training

3.3.1 Model and Framework

Whisper [RWKX+22] is a family of models for multilingual speech recognition and transcription created by OpenAI in 2022. Their training data consisted of on over 600 thousand hours of multilingual audio collected from the internet. Each model works with 30-seconds segments of audio, converted into spectrograms and encoded. As of 2025, the whisper models are available in 6 sizes, each one larger and more accurate than the previous one. However, larger models also require more compute time and have higher memory usage [Whi]. Table 3.3.1 below shows the available models and each one's number of parameters.

Model	**Number of Parameters**
tiny	39M
base	74M
small	244M
medium	769M
large	1550M
turbo	809M

Table 3.6: Model sizes according to [Whi]

Owing to limited resources, compute units and storage, the **whisper-small** model was the one used for the experiment. The Hugging Face (**HF**) suite of libraries provided support for generating the required datasets for fine-tuning the base model through the **Datasets** library [Huga], for training through the **Transformers** library [Hugd] and for fine-tuning through the Parameteter-Efficient Fine-Tuning (**PEFT**) library [Hugc].

3.3.2 Preparation

The data is preprocessed and loaded via a custom Loader class. This class is used to load Common Voice data, but a common interface could be defined to load data from other sources, such as USPDATRO. The loader is customizable via two parameters referring to the amount of Italian and Spanish data relative to the amount of Romanian data. Foreign transcriptions are processed through the previously defined pipeline. The Loader also handles the caching of the downloaded data so as to avoid processing the same transcription twice and assures that the foreign data is only used in the training dataset, keeping the testing and validation sets purely Romanian.

The audio is resampled to 16kHz in the case when it is not so by default and passed to the Whisper processor.

3.4 Measuring Results

3.4.1 Word error rate

Word-error-rate **(WER)** emerges as the metric most frequently used in similar studies. The formula used for computing WER is based on the Levenshtein distance between the expected and transcribed phrases: total number of substitutions, deletions, and insertions required for turning the generated transcriptions into the correct ones, all divided by the total number of words. Neither punctuation, nor capitalization are relevant to the task. As such, both the input and the output texts have their characters mapped to lower case and their punctuation removed.

$$WER(t, m) = \frac{S_W(t, m) + D_W(t, m) + I_W(t, m)}{N_W(t)}$$

where
$t :=$ transcription
$m :=$ model's output
$S_W(t, m) :=$ word substitutions required to turn m into t
$D_W(t, m) :=$ word deletions required to turn m into t
$I_W(t, m) :=$ word insertions required to turn m into t
$N_W(t) :=$ word count of t
Although generally ranging between 0 and 1, it is possible for the WER to be greater than 1 in the case when the model's output has more words than the transcription. For example,

$$WER("cineva", "cine\ va") = \frac{1 + 1 + 0}{1} = 2.0$$

$S_W("cineva", "cine\ va") = 1$, since "cine" must be substituted with "cineva"
$D_W("cineva", "cine\ va") = 1$, since "va" must be deleted
$I_W("cineva", "cine\ va") = 0$, since no other words must be inserted

3.4.2 Character error rate

Analogous to the WER, Character-error-rate **(CER)** calculates similar values, but on a per-character basis. This is particularly important in the context of Romanian, where small differences - such as an "i" instead of an "e" between a "c" and an "a" - occur frequently. In the case when two models yield a very similar WER, the CER can serve as a tiebreaker.

$$CER(t,m) = \frac{S_C(t,m) + D_C(t,m) + I_C(t,m)}{N_C(t)}$$

where
$t :=$ transcription
$m :=$ model's output
$S_C(t,m) :=$ character substitutions required to turn m into t
$D_C(t,m) :=$ character deletions required to turn m into t
$I_C(t,m) :=$ character insertions required to turn m into t
$N_C(t) :=$ character count of t

3.4.3 Relative error rates

By choosing a baseline model and measuring its performance in parallel with the fine-tuned models, two new metrics can be defined: Relative Word Error Rate (**RWER**) and Relative Character Error Rate (**RCER**).

$$RWER_b(t,m) = \frac{WER(t,m)}{WER(t,m_b)}$$

$$RCER_b(t,m) = \frac{CER(t,m)}{CER(t,m_b)}$$

where
$t :=$ transcription
$m :=$ model's output
$b :=$ baseline model
$m_b :=$ baseline model's output

Seeing as the purpose of this paper is to assess the usefulness of incorporating speech data from multiple languages, the chosen baseline model will be the pre-trained `openai/whisper-small`, the same as the base used for training. By measuring $RWER_{whisper-small}$ and $RCER_{whisper-small}$ for each of the fine-tunes, it is possible to assess if the fine-tuning process lead to an increase in accuracy. A relative error rate will always be positive, since WER and CER both produce positive results. As such, a relative error rate, be it word, or character-based, `r`, will belong to one of the following partitions:

- $r = 0$: The model had an absolute error rate of 0, it transcribed every audio snippet perfectly; It might be that the model performed some sort of overfitting;
- $0 < r < 1$: The model outperformed the baseline; Depending on how good the baseline model was, this can be categorized as a successful fine-tuning process;
- $r \geq 1$: This model had an identical or worse performance than the baseline model; This can be categorized as an unsuccessful fine-tuning process;

3.5 Constructed Datasets

For the purpose of this experiment, we are interested in measuring the relative performance of models fine-tuned on different amounts of data generated with the pipeline described in by previous section. As such, 12 datasets were created based on CommonVoice. Each dataset was characterized by 3 parameters: the number of Romanian ("pure") samples, the number of Italian samples as a percentage of the Romanian number, and the number of Spanish samples as a percentage of the Romanian number. The samples originating from foreign languages are only included in the training set, keeping the testing and validation splits "pure". The Romanian Corpus of CommonVoice had an 80-10-10 split, for training, validation and testing, respectively. This resulted in 4000 samples in the training set, and 500 samples in the validation and testing sets each. The same validation and testing sets were used across all 12 datasets, meaning that testing and validation will happen on the same samples, regardless of model.

A cache system was put in place in order to ensure no sample from the Italian and Spanish CV datasets is processed twice. In practice, this means that when creating a dataset with `n` Italian samples for the first time, `n` samples will be transformed using the pipeline. Requesting `n + m` samples after that would reuse the previous `n` samples and only process the remaining `m` ones. As a byproduct,

$$\text{Dataset}_{\substack{\text{it}=x_1\\ \text{sp}=y_1}} \subseteq \text{Dataset}_{\substack{\text{it}=x_2\\ \text{sp}=y_2}} \Leftrightarrow x_1 \leq x_2 \text{ and } y_1 \leq y_2$$

Table 3.5 shows the 12 generated datasets, alongside the size of each one's training set. Calculating the size of each training set can be done with the following formula:

$$4000 * (\frac{ItFr}{100} + \frac{SpFr}{100}) = 40 * (ItFr + SpFr)$$

Dataset Name	Italian Fraction	Spanish Fraction	Training Set Size
dataset-5k-00it-00sp	0	0	4000
dataset-5k-05it-05sp	5	5	4400
dataset-5k-15it-15sp	15	15	5200
dataset-5k-25it-25sp	25	25	6000
dataset-5k-35it-35sp	35	35	6800
dataset-5k-50it-50sp	50	50	8000
dataset-5k-50it-00sp	50	0	6000
dataset-5k-00it-50sp	0	50	6000
dataset-5k-05it-25sp	5	25	5200
dataset-5k-25it-05sp	25	5	5200
dataset-5k-35it-15sp	35	15	6000
dataset-5k-15it-35sp	15	35	6000

Table 3.7: Dataset Sizes

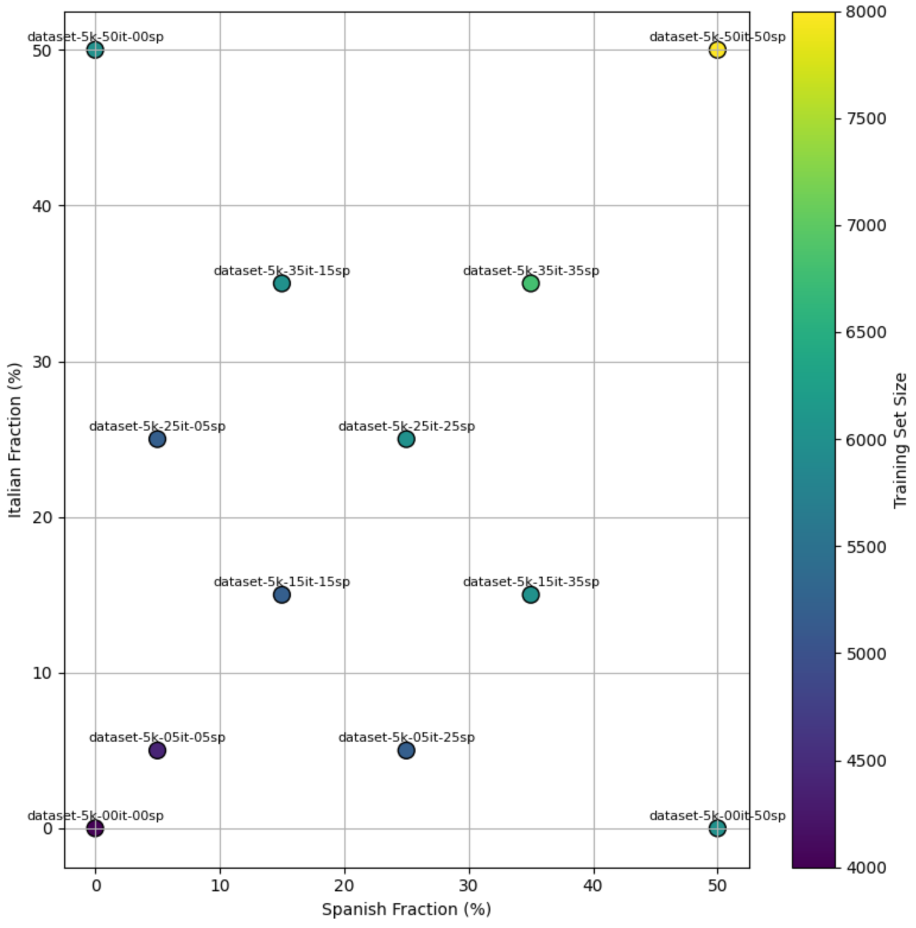


Figure 3.2: Datasets are arranged according to Italian and Spanish percentages

The generated datasets are clustered around the lower left corner corner of the matrix. Based on the findings of [Zga19] and [GP23], augmenting a dataset with foreign data leads to worse results when the percentage of foreign data becomes overbearing. This is the reason why none of the generated datasets have Italian or Spanish fractions going over 50%. It should be noted that all entries situated above the first diagonal have more than a third of the data from a foreign origin. The generated datasets are publicly available on HuggingFace, at the links in table 3.5.

3.6 Model Training

Based on each generated dataset, a version of `whisper-small` was fine-tuned inside a Google Colab **A100 environment** using a custom script based on HuggingFace's Transformers [Hugd] and PEFT [Hugc] libraries. To enable efficient fine-tuning while minimizing the number of trainable parameters, parameter-efficient fine-tuning (PEFT) techniques were applied. The model was **initialized** from the `whisper-small` checkpoint. **Decoder constraints** were relaxed by setting emptying `forced_decoder_ids` and `suppress_tokens`. **PEFT** was applied with dimension for adapter layers set to 32, scaling factor set to 64, adapters applied only to the **query and value projection** layers of the attention mechanism and the dropout rate set to 0.05.

The training was conducted over 5 epochs with a batch size of 32 per device, **gradient accumulation** over 2 steps, learning rate of 2×10^{-5} , 30 warm-up steps, mixed-precision training enabled and metrics computed using `wer` and `cer`.

The validation and test splits were the same, regardless of dataset used, ensuring a common metric for all models. Training was achieved using the Seq2SeqTrainer interface, with a task-specific data collator and generation-based metric computation.

Model	Dataset
finetune-5k-00-00	dataset-5k-00it-00sp
finetune-5k-05-05	dataset-5k-05it-05sp
finetune-5k-15-15	dataset-5k-15it-15sp
finetune-5k-25-25	dataset-5k-25it-25sp
finetune-5k-35-35	dataset-5k-35it-35sp
finetune-5k-50-50	dataset-5k-50it-50sp
finetune-5k-50-00	dataset-5k-50it-00sp
finetune-5k-00-50	dataset-5k-00it-50sp
finetune-5k-05-25	dataset-5k-05it-25sp
finetune-5k-25-05	dataset-5k-25it-05sp
finetune-5k-35-15	dataset-5k-35it-15sp
finetune-5k-15-35	dataset-5k-15it-35sp

Table 3.8: Models trained for each dataset

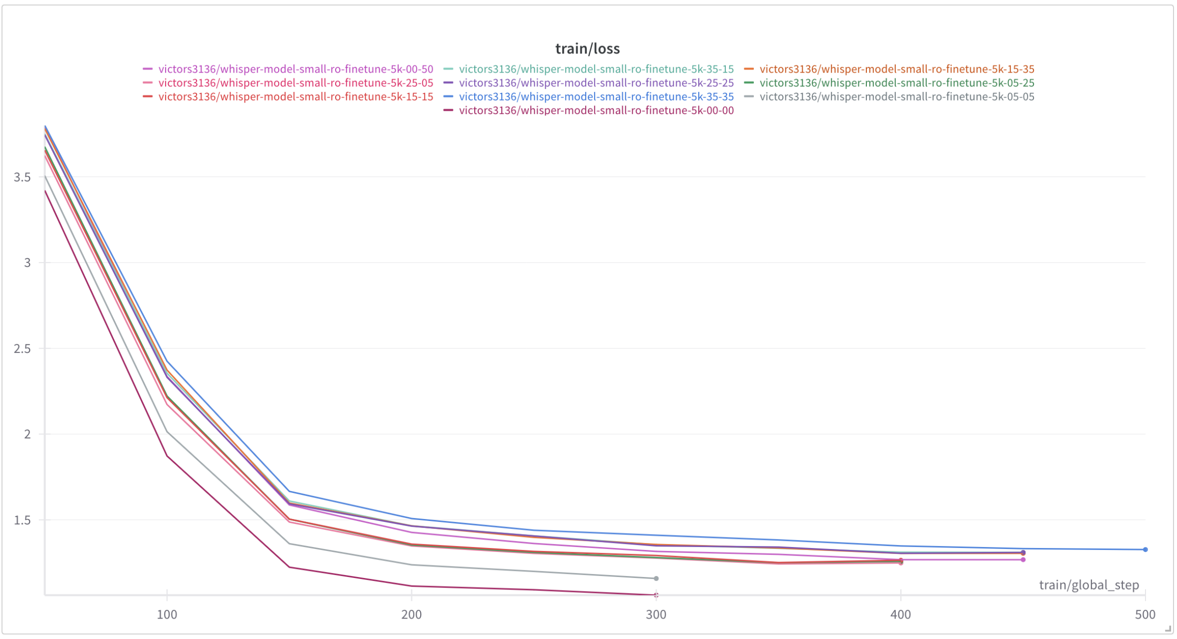


Figure 3.3: Training Loss of 10 of the 12 models

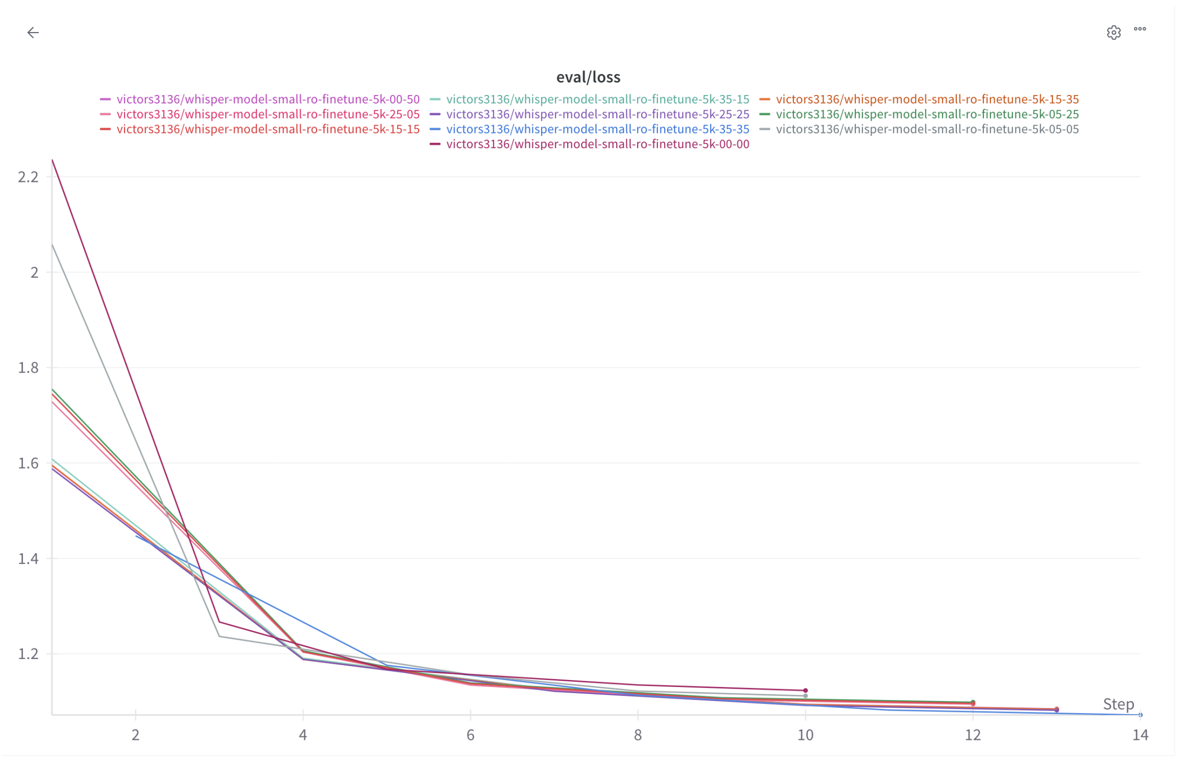


Figure 3.4: Validation Loss of 10 of the 12 models

Figures 3.3 and 3.4 show the training and validation loss for 10 of the 12 models, as reported by the trainer to Weights and Biases. The amount of steps varies due to the differing training set sizes and a fixed epoch count. All models exhibit a steep decline in loss within the first 100 training steps, followed by gradual convergence. Models trained with smaller proportions of foreign-language data tend to converge faster and reach lower final loss values, suggesting better fit on Romanian-only validation data, which is expected given that they have less noise in the dataset.

3.7 Evaluation

As discussed during Section 1 of this chapter, 500 entries of the Common Voice dataset were not included in any of the models' training or validation sets. In order to assess their performance, the baseline model, `whisper-small`, also benchmarked against the same dataset. WER and CER were computed for each individual model. RWER and RCER were computed with respect to `whisper-small`'s metrics.

3.8 Overview

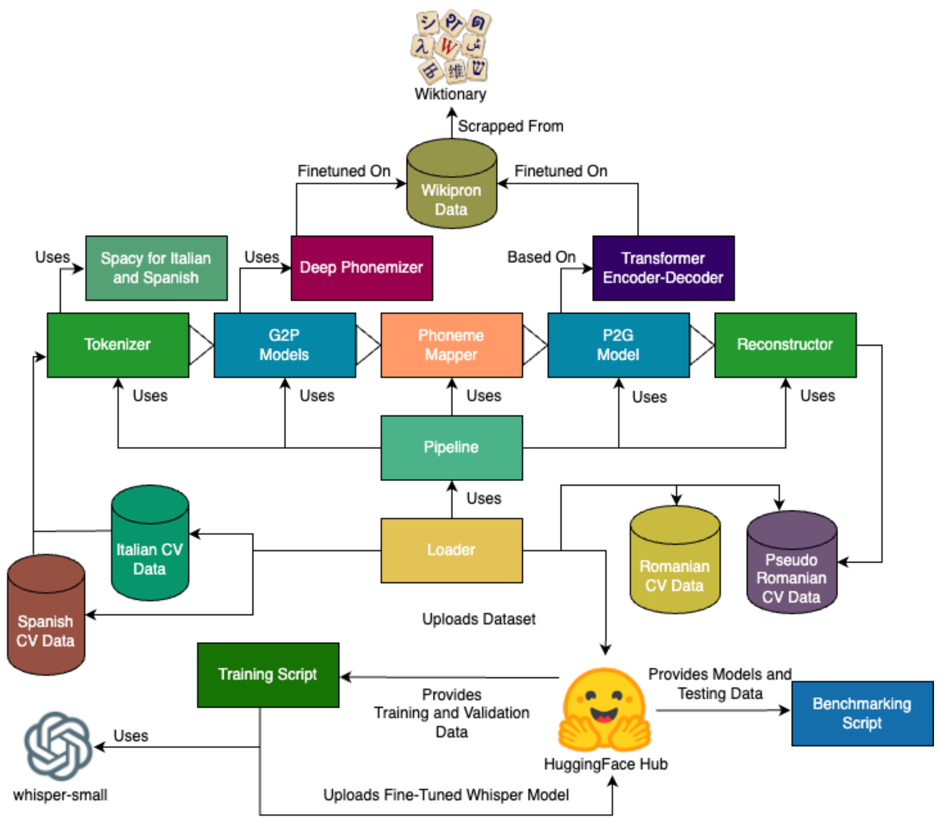


Figure 3.5: Training a Model

Figure 3.5 shows the entire structure of the project. The architecture from Figure 3.1 is encapsulated by the Pipeline class at the center of the current figure. The pipeline makes use of the Common Voice Italian and Spanish datasets, which get transformed through an initial Spacy-based tokenization. Data then passes through a version of Deep Phonemizer [spr], fine-tuned using data scraped from Wiktionary using the Wikipron library[CC]. The mappings between source and target language are the ones described in section 3.2.2. After undergoing phonetic conversion, data is passed to an encoder-decoder model, fine-tuned with Wikipron data as well. The last step involves word and phrase reconstruction, restoring whitespace and punctuation. The data used for fine-tuning the Italian and Spanish G2P models, as well as the Romanian P2G model is available at the following link: `https://github.com/victors3136/Training-Data`. The result of the processing pipeline is represented by Pseudo Romanian CV Data. The Loader class is responsible for combining that data with Romanian CV and pushing the result to Hugging Face Hub.

3.9 Results and Interpretations

The results, presented in table 3.9, show that half of the 12 trained models outperformed the baseline one. For the successful models, there seems to be no difference if ranking is done according to either metric, but there is a difference for the unsuccessful ones.

Model ID	WER	Relative WER	CER	Relative CER
finetune-5k-00-50	2.96	0.65	2.01	0.75
finetune-5k-00-00	3.31	0.73	2.06	0.77
finetune-5k-05-25	3.39	0.75	2.15	0.80
finetune-5k-05-05	3.47	0.77	2.22	0.83
finetune-5k-15-35	3.88	0.86	2.39	0.90
finetune-5k-15-15	4.11	0.91	2.43	0.91
whisper-small	4.49	1	2.66	1
finetune-5k-35-15	4.79	1.06	2.86	1.07
finetune-5k-25-05	4.79	1.06	2.75	1.03
finetune-5k-25-25	4.93	1.09	2.82	1.06
finetune-5k-50-50	5.35	1.19	3.15	1.18
finetune-5k-35-35	5.58	1.24	3.18	1.19
finetune-5k-50-00	5.89	1.31	3.11	1.17

Table 3.9: Performance metrics for fine-tuned models, truncated to 2 decimals

Metric	Mean $\bar{x}$	Standard Deviation s
Relative WER	0.97	0.2
Relative CER	0.97	0.14

Table 3.10: Statistical measures of metrics used

Table 3.10 shows the mean and standard deviation of RWER and RCER over the entire population of 13 benchmarked models. On average, the models performed better than the baseline. The small standard deviation indicates a relatively low variance between models, which is to be expected since the models were all fine-tuned on the same core Romanian dataset.

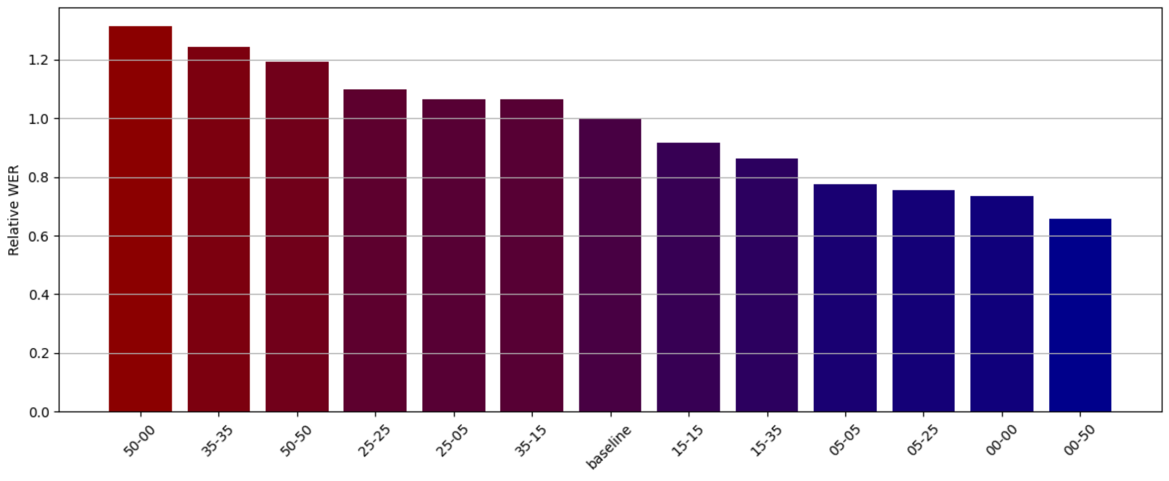


Figure 3.6: Relative WER

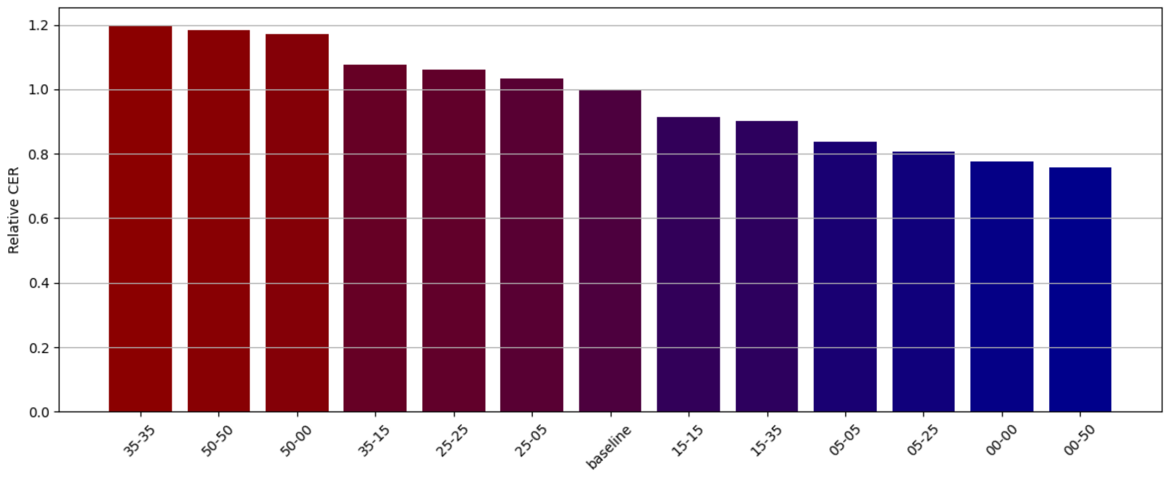


Figure 3.7: Relative CER

Figures 3.6 and 3.7 show the models arranged in descending order based on each respective metric. The best overall model is `finetune-5k-00-50`, outperforming both the baseline `whisper-small` and `finetune-5k-00-00` (corresponding to a normal fine-tuning process, not involving any foreign data).

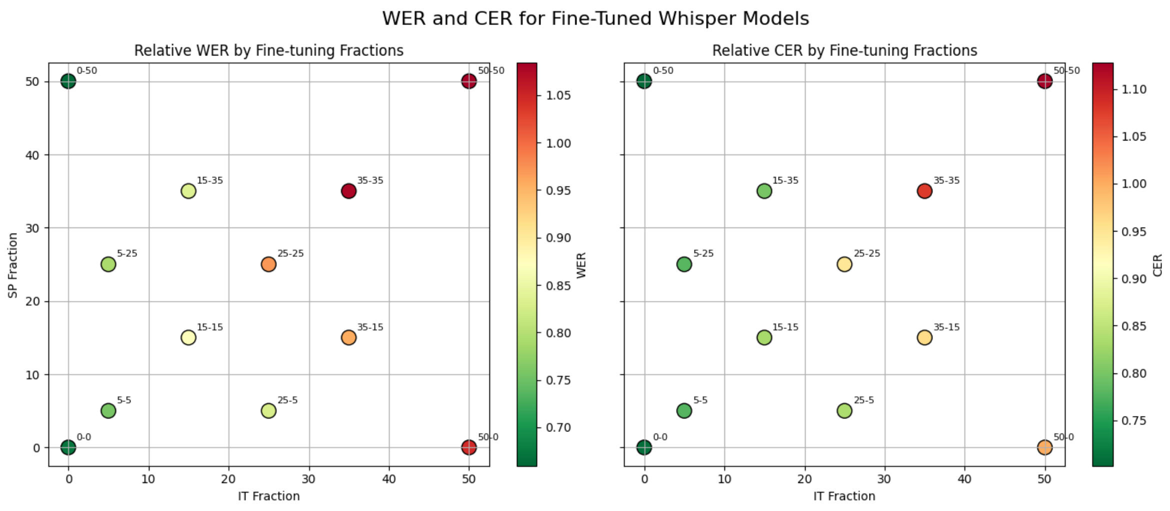


Figure 3.8: Results of the 12 trained models

Figure 3.8 shows the 12 fine-tuned models plotted on a cartesian coordinate system, with the abscissa representing the fraction of Italian in the original dataset and the ordinate representing the Spanish fraction. Relative error rates are represented by the color of each dot, with green and yellow dots indicating the models which outperformed the baseline. The best performing models are dark green, while the worst performing ones are dark red.

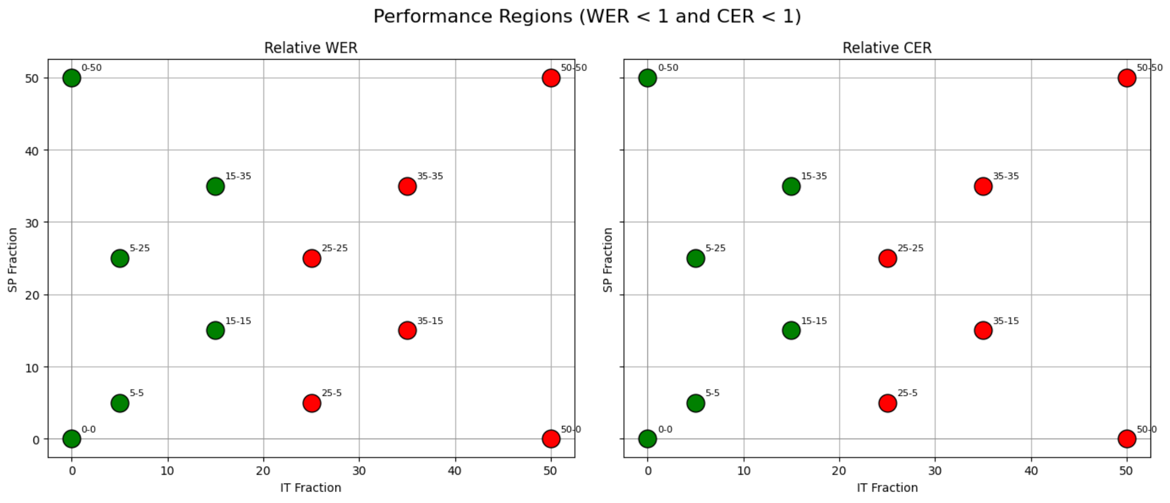


Figure 3.9: Models colored according to whether or not they outperform the baseline

Figure 3.9 displays the models on the same Italian and Spanish axes as in 3.8 with the coloration modified to show which ones outperformed the baseline. Through this figure, 2 distinct regions can be clearly distinguished: `it < 25` and `it` $\geq$ `25`. Italian samples seem to have an overbearing negative result over the resulting models. Figures 3.10 and 3.11 show relative error rates plotted against the fraction of Italian and Spanish. In both figures, Italian fraction is almost inversely proportional to the model's performance. On the other hand, the same figures show Spanish fractions having a beneficial effect up to a certain degree. The models `finetune-5k-15-35` and `05-25` performed better than `15-15` and `05-05`, respectively.

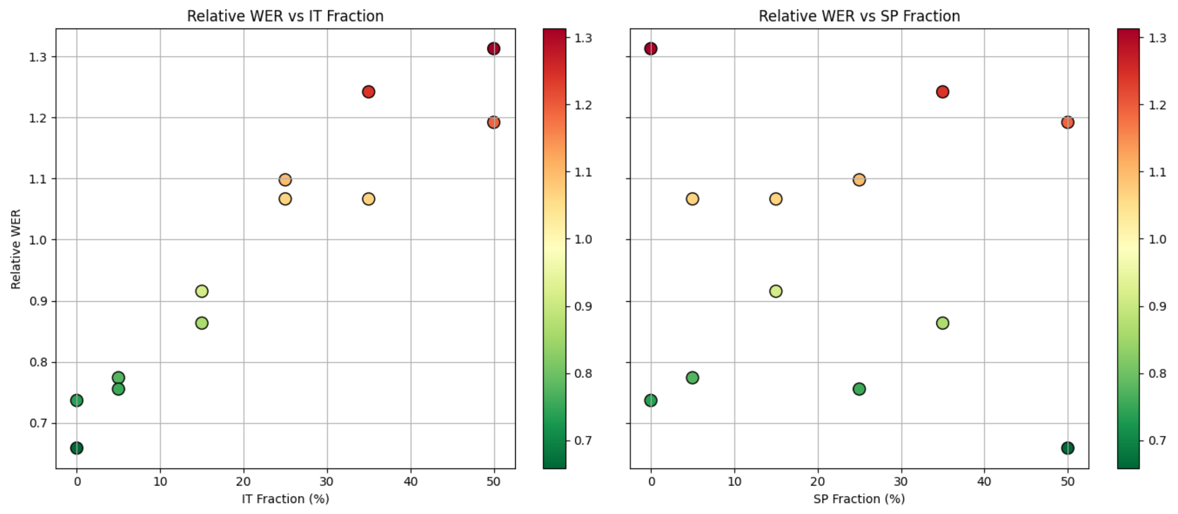


Figure 3.10: RWER plotted against Italian and Spanish fractions

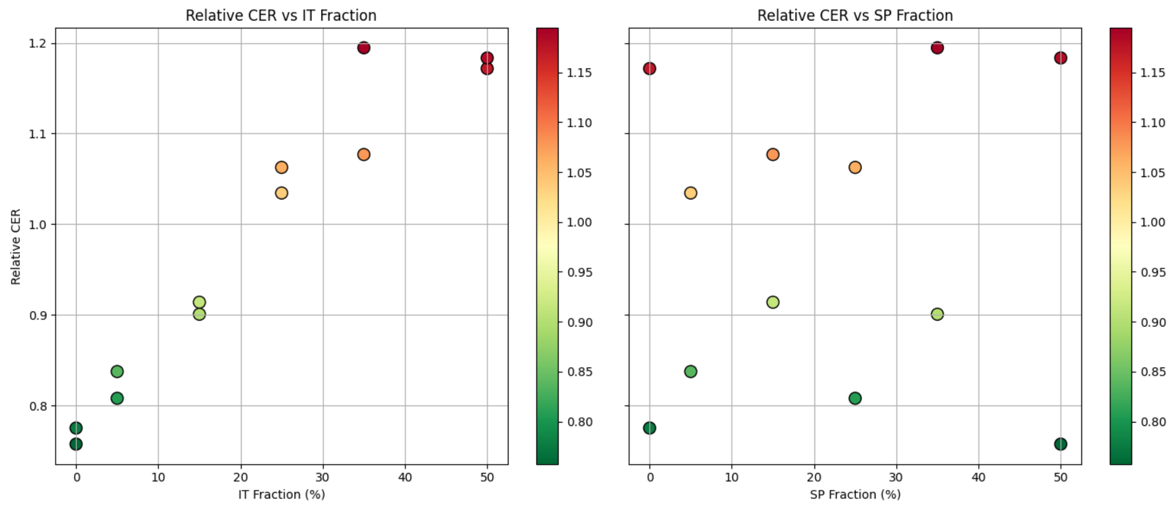


Figure 3.11: RCER plotted against Italian and Spanish fractions

The negative results associated to the Italian percentage of data are unexpected. Searching for a reason may lead to one of the following conclusions. First off, Italian and Romanian speech patterns may simply be incompatible. This would contradict the findings of [GP23] and even [CBC+21] to a certain degree. Another cause may be a failure of the data generation pipeline. If that were the case, the failure must have happened before the point where the two pipelines join. That seems improbable, but not impossible, as the steps for Italian and Spanish were developed in parallel. The failure could be owed to the similarity between Spanish and Italian. Perhaps, with the two of them being so similar in written form, but quite different phonetically, they cause the model to be unable to properly generalize. This also cannot be true, as the performance of the `it = 50, sp = 0` model is among the worst ones. As a final possibility, it may be that there are subtle differences between Common Voice datasets. Factors such as recording equipment, speaker tone, and background sounds could be introducing too much noise for `whisper-small` to generalize.

It could be useful to use larger models and more diverse datasets in the future to assess the validity of each of these factors.

Following the experiment it is possible to return to the initial research questions:

Answer for Question 1: 1.1 The results of such a process are mixed and depend a lot on the proportion of source languages used and on how well the audio samples of the source languages match the ones of the target language. Done in moderation, this process can still enhance the performance of the final system.

Answer for Question 2: 1.1 The limit discovered by Gasan and Păiș [GP23], of about 30%, seems to hold, even when the data comes from multiple source languages.

Answer for Question 3: 1.1 Due to the suspected error in the Italian samples, no response can be reliably presented in this matter.

Answer for Question 4: 1.1 Comparing `it = 50, sp = 0` and it = 50, sp = 50, it seems like Spanish samples may, in fact, mitigate the failure of Italian samples.

Chapter 4

Application

This chapter describes the final application architecture used to serve our most representative ASR model and expose its capabilities to users via a client interface.

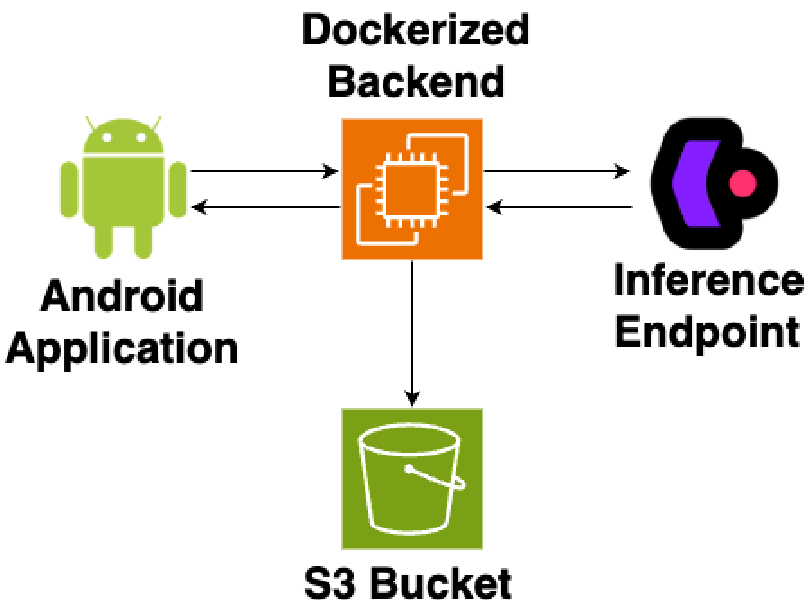


Figure 4.1: Overview of Application

The application illustrated in Figure 4.1 exposes one of the best performing models, `finetune-5k-15-35`, to users via an Android application. The model was chosen for being the model with the most source language data of the successful ones. The model's inferences are served through the Inference Endpoints, a service provided by Hugging Face which supports transforming AI models into APIs [Hugb]. A backend service is set up to handle the uploading of files from consenting users of the application to an S3 bucket, allowing for further developing tools for Romanian ASR systems.

4.1 Client

4.1.1 Requirements

The client is a lightweight mobile application designed to interface with the ASR server. Its main requirements are as follows:

- Capture audio from the microphone for at most 30s;
- Send audio to the ASR server and receive transcription;
- Display the transcript in a user-friendly interface;
- Display any possible errors in a user-friendly interface;
- Record user information such as gender and age for further development of Romanian ASR tools;
- Allow users to opt out of data collection;
- Support different screen sizes and color themes;

Figure 4.2 describes the application from the point of view of the end user. When a user enters the application, he should be able to record an audio file of at most 30s by pressing a simple button. While a recording is active, he should have the option to cancel the recording or to submit it to the server for analysis. While a recording is not active, the user should be able to enter a form by pressing a designated button.

The form should load data from the user preferences if such data is available, otherwise should display default values for every field. The form should have the following fields: age (accepting only numerical inputs), gender (accepting one of the following: "woman", "man", "other"), and consent (a simple checkbox). The default value for user consent will be false.

Regardless of the current screen, a button for changing the application theme should always be available. Changing the theme by pressing said button should not interfere with any ongoing operation (recording or uploading).

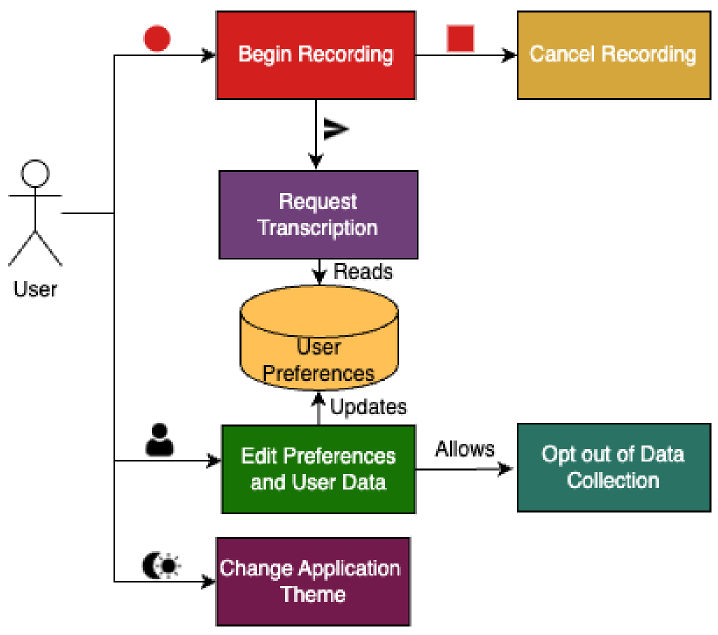


Figure 4.2: Front End Functionalities

4.1.2 Implementation

The client is implemented in Kotlin for Android, using the Jetpack Compose framework. Interfacing with the server is done via the Okhhtp library [OKH] and the icons are provided by Google's Material Design 3 library [Goo].

The application provides 2 screens:

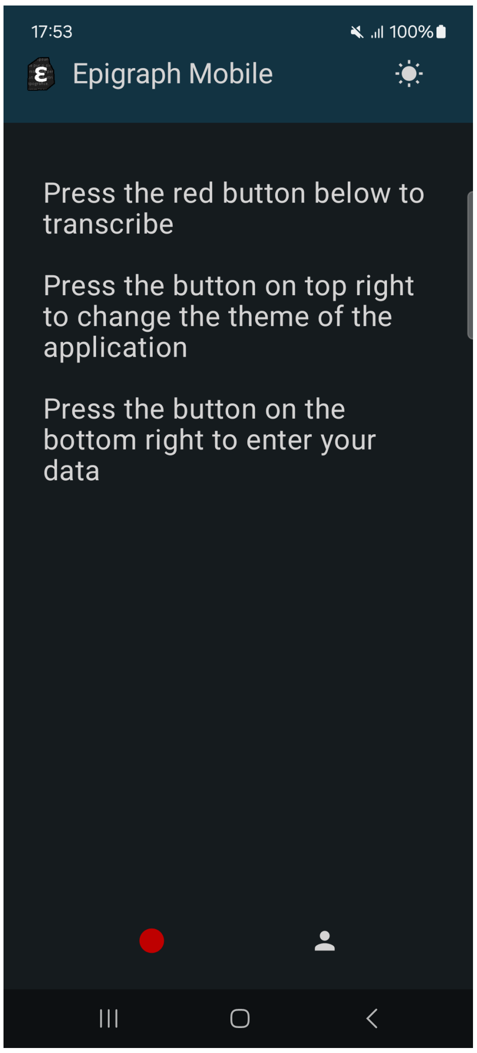
- Audio recording screen;
- User prefernces from screen;

Audio Recording Screen

This is the screen that the user is welcomed by. By default, it displays a quick guide on what each button on the screen does. It provides 2 buttons in its footer: one for starting the recording and one for editing user preferences.

When pressing the "Record" button represented by the red dot, the text on screen is replaced with an animation showing some sound bars and the current duration of the recording. The buttons on the bottom are replaced with buttons for canceling or uploading the recording. Pressing the "Stop" button represented by the red square return the page to the original welcome message.

When submitting the audio file, the content of the page is replaced with the response from the server. In case of an error, the message displayed has a red color, otherwise it is displayed in the same color as all the rest of the application's text.


17:53

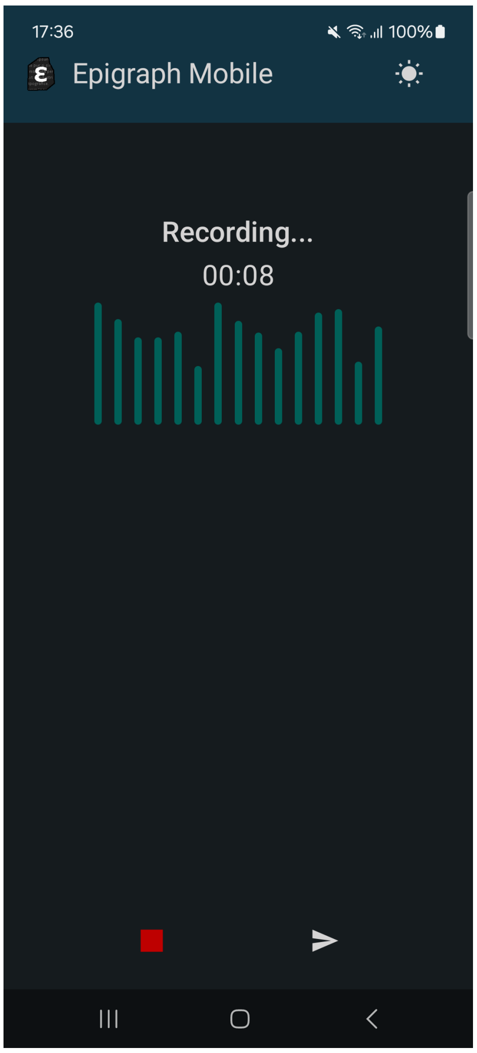
Epigraph Mobile

Press the red button below to transcribe

Press the button on top right to change the theme of the application

Press the button on the bottom right to enter your data

Figure 4.3: Welcome message


17:36

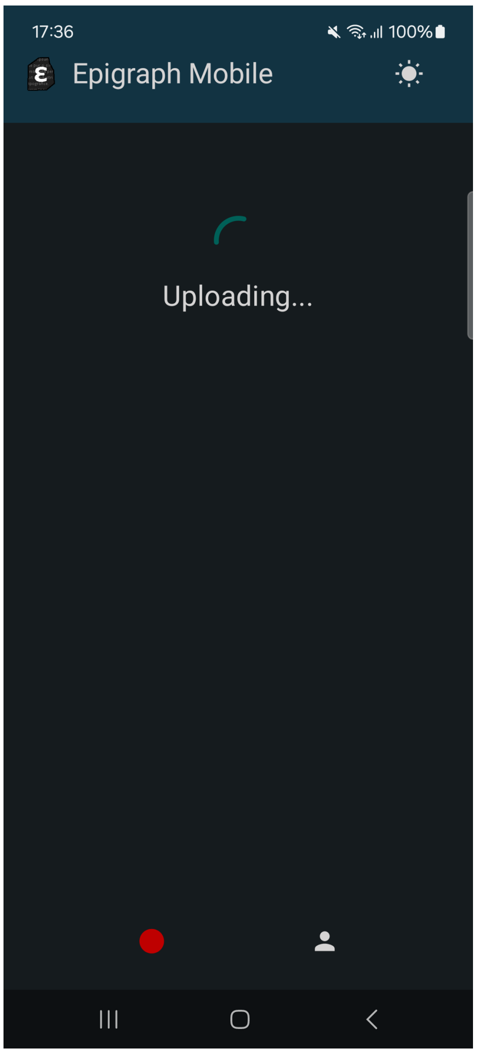
Epigraph Mobile

Recording...

00:08

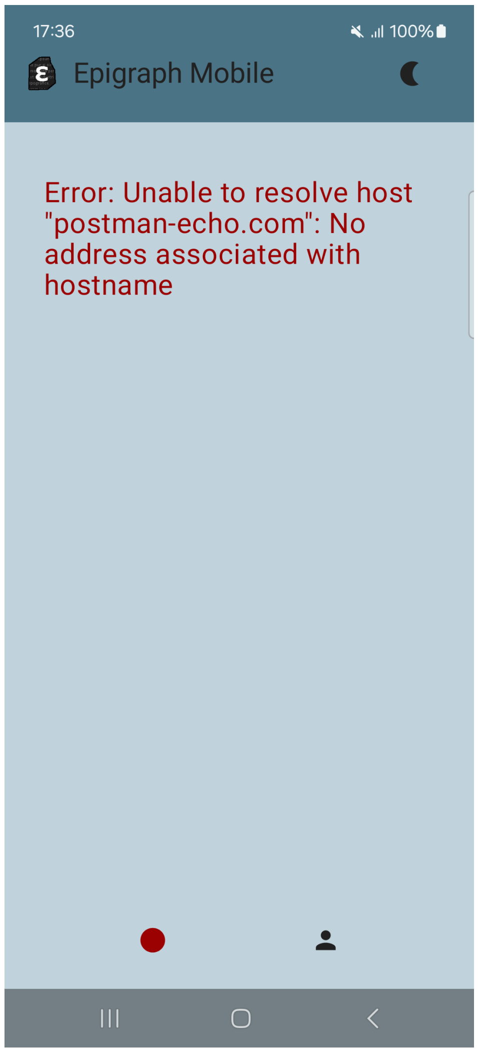
Figure 4.4: Still frame from the recording animation

While waiting for the response from the server, the application displays a loading animation.



Uploading...

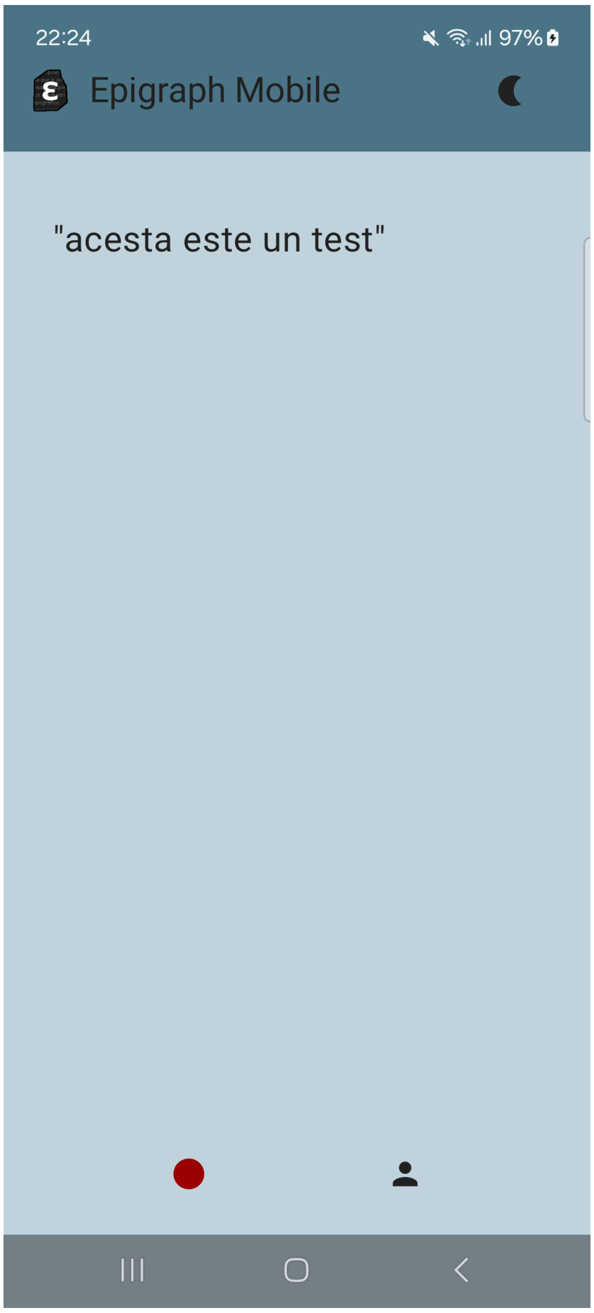
Figure 4.5: Still frame from the loading animation



Error: Unable to resolve host "postman-echo.com": No address associated with hostname

Figure 4.6: Error message when no internet is available

The final, successful, response should be the transcription received from the server. From this view, the user should still have access to the initial buttons, the one for beginning recording and the one for editing user preferences.



"acesta este un test"

Figure 4.7: Transcription from the server

User Preferences Form Screen

This screen allows the users to specify data relevant to themselves and the audio file they provided. It is here also that they may opt in or out of allowing the server to upload their audio files to the S3 bucket.

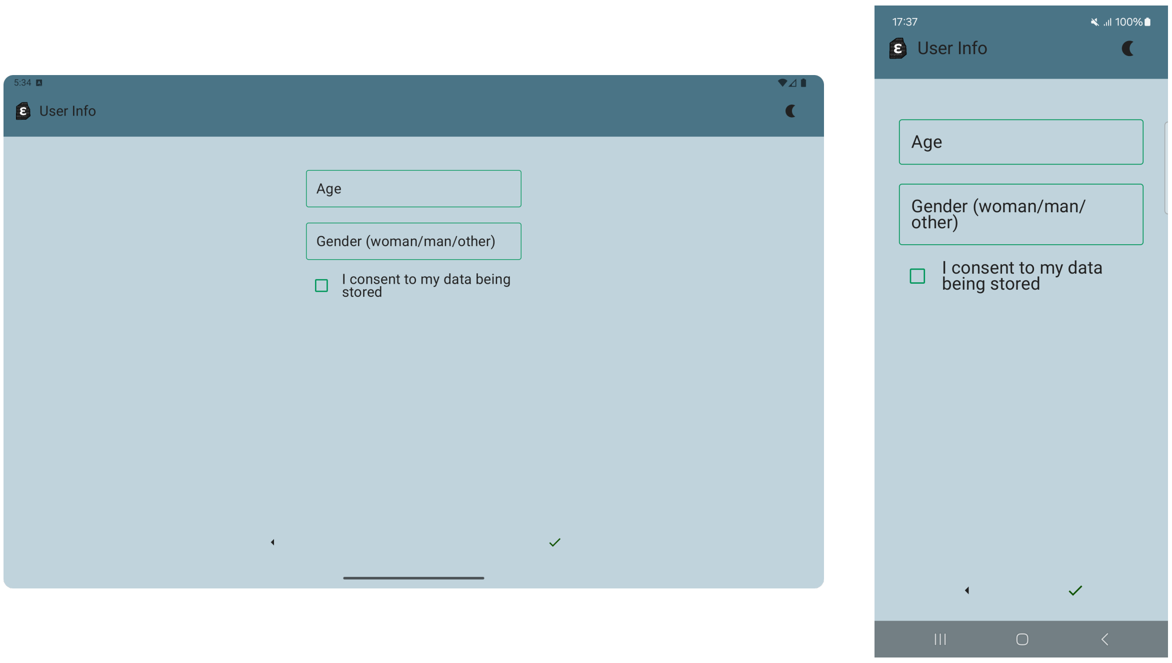


Figure 4.8: User preferences from available in 2 different screen orientations, on 2 different devices

The age field is constrained to only numerical values, while the gender field is constrained to the values "woman", "man" and "other". When submitting the form, a validation step assures that age and gender are valid values. The form must be submittable even if the user did not give out his consent for data collection.

When pressing the submit button, if the data is valid, the user is taken back to the main screen. If it is not, the user receives a toast notification warning them of the inconsistencies.

Data is saved to and loaded from a data store. This data store is configurable in order to allow isolated testing of the save and load functionalities.

4.1.3 Testing

In order to verify that the application meets all of its requirements, testing focused on the following areas:

User preferences storing and retrieval

A mock data store was created for the purpose isolated of testing and the functions' signatures were updated to allow context-dependent data stores.

- **Unit testing for writing data:** different age, gender, and consent values were chosen, ensuring the data is correctly written down in the data store;

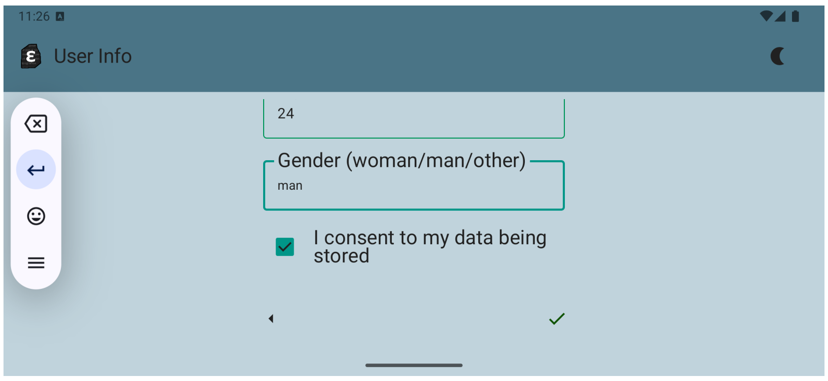
- **Unit testing for reading data:** different age, gender, and consent values were chosen, ensuring the data is correctly retrieved from the data store;
- **Integration testing:** sequences of read and write operations were tested in order to ensure correct behavior in the following scenarios:
  - read (before any write);
  - write -> read;
  - write -> overwrite -> read;

User Interface

Compatibility testing was conducted on multiple platforms, with differing Android versions, screen sizes, and orientations, ensuring that the user interface remains functional regardless of device, as illustrated by Figure 4.9.

The devices used for verifying were:

- Physical Items:
  - Samsung A51 running Android 13;
  - Samsung A53 running Android 14;
- Virtual Devices (provided by Android Studio):
  - Google Pixel Phone running Android 15;
  - Google Pixel Tablet running Android 16;



24

Gender (woman/man/other)
man

I consent to my data being stored

Figure 4.9: Horizontal form layout for Google Pixel with Android 15

4.1.4 Deployment

The latest APK is available for download on the following GitHub page and on the server's main page, which will be detailed in the subsequent section.

4.2 Server

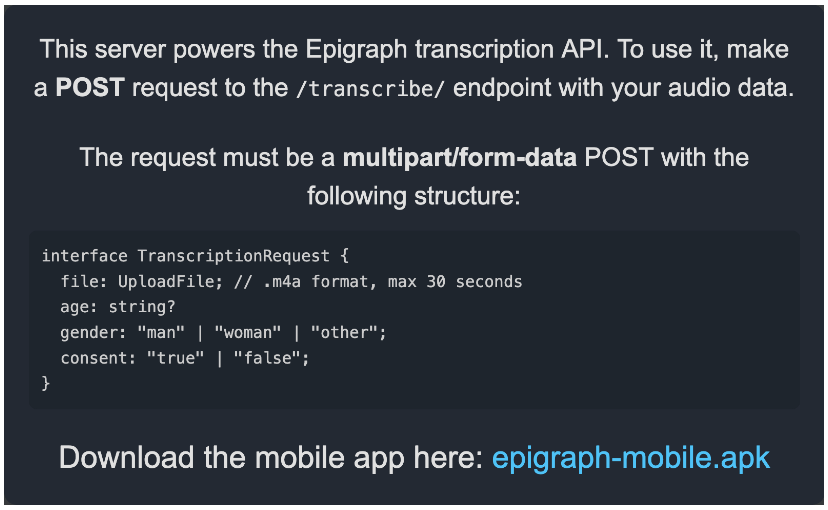
4.2.1 Requirements

The server is responsible for exposing the trained ASR model via a REST API. The core requirements for this component are as follows:

- Serve requests over HTTP via a RESTful interface;
- Accept audio files as input and return transcriptions;
- Forward requests to the Inference Endpoint, making sure that the model receives the input data in a `.wav` format;
- Keep no state in order to allow horizontal scaling;
- Store and categorize recorded data received from clients;

4.2.2 Implementation

The server uses the FastAPI framework [Ram] in order to expose 2 endpoints. The first one, `GET /`, returns an HTML page providing the user with a quick guide to using the second endpoint and a link to download the client Android application. The page can be seen in Figure 4.10.


This server powers the Epigraph transcription API. To use it, make a **POST** request to the `/transcribe/` endpoint with your audio data.

The request must be a **multipart/form-data** POST with the following structure:

```
interface TranscriptionRequest {
  file: UploadFile; // .m4a format, max 30 seconds
  age: string?
  gender: "man" | "woman" | "other";
  consent: "true" | "false";
}
```

Download the mobile app here: epigraph-mobile.apk

Figure 4.10: Server Home Page

The second endpoint, as described by 4.10, is `POST /transcribe/`. It expects to receive a file in `m4a` format and some metadata. If the user consents, the file and its metadata are uploaded in an s3 bucket. After that, the audio file gets converted to `.wav` and forwarded to the inference endpoint. The endpoint's transcription is then returned.

Metadata is validated on the server side to ensure proper values. The file contents are also validated server side, ensuring the file is in the correct format and has at most 30 seconds. Manipulation of audio data is done through the pydub library [Rob]. During development, the FFmpeg framework [Tea] was used to convert files `mp4` to `wav` and to generate `mp4` files for testing.

Amazon's S3, or the Simple Storage Service, is a service offered as part of Amazon Web Serivces' suite. S3 allows users to store any type of objects up to 5TB in size. The pricing is based on usage, therefore price alerts were set up in preparation for the application's deployment. The S3 bucket was set up in the region `eu-north-1`. Its aim is to store the audio files sent by the application's users, provided that they gave their consent. In time, the bucket "romanian-asr-data" can provide a useful base of knowledge for future Romanian ASR projects. Interfacing with the S3 bucket was achieved through the use of the boto3 library [AWS], provided as a part of AWS' SDK for Python.

4.2.3 Testing

Unit testing was performed in order to ensure all utilities made for input data validation and sanitation. Among the functionalities tested, the most noteworthy ones are converting an mp4 to a wav, finding the duration of an audio file, and metadata validation.

4.2.4 Deployment

The server was deployed using another one of AWS' services. Belonging to the Infrastructure-as-a-Service family, EC2, or Elastic Compute Cloud, offers virtual machines that can be configured to be publicly accessible from the wider internet. The server was containerized using Docker [Doc] in order to allow for simple horizontal scaling. The server is publicly accessible at the url:

`http://ec2-13-60-99-27.eu-north-1.compute.amazonaws.com:8000/`.

Chapter 5

Conclusions

In conclusion, this paper presented a new way to train better ASR systems for languages with limited tooling and resources. It chronicled the steps for developing a lightweight and quick to train automatic speech recognition application for the Romanian language, from dataset creation, to model building and web deployment. During the ASR model's development, a suite of open source tools were used, of which Mozilla's Common Voice Datasets and Hugging Face's Transformers library proved to be of immense help. The model was fine-tuned from OpenAI's `whisper-small` model using Romanian data from Common Voice, augmented with Italian and Spanish data, transformed to align with Romanian phonetic rules.

The findings show that cross linguistic speech recognition using multiple source languages is possible and can have positive results when data for the target language is limited. It was shown that Spanish is more useful than Italian for this purpose, despite the expectation that Italian would align more closely to Romanian. It was also shown that better voice recognition systems for Romanian can be achieved using low-cost fine-tuning and multilingual data.

Each of the datasets and models created for the purpose of this paper are publicly available. A storage of Romanian audio data was set up and can be filled with recordings from consenting users of a mobile application. An extensible and flexible set of modules was created for generating cross linguistic datasets.

While the results seem promising, several limitations remain. In the future, this experiment could be recreated with larger resource pools and targeted experimental design. Based on the current findings, it is possible to focus on less model configurations from the very start. Additionally, data collected through the final application can be used to continually develop better tooling for Romanian.

Bibliography

[ABD^{+}20] Rosana Ardila, Megan Branson, Kelly Davis, Michael Henretty, Michael Kohler, Josh Meyer, Reuben Morais, Lindsay Saunders, Francis M. Tyers, and Gregor Weber. Common voice: A massively-multilingual speech corpus. *Proceedings of the Twelfth Language Resources and Evaluation Conference*, 2020.

[APT20] Andrei-Marius Avram, Vasile Păiș, and Dan Tufiș. Towards a romanian end-to-end automatic speech recognition based on deepspeech2. In *Proceedings of the Romanian Academy, Series A*, pages 395–402, 2020.

[Ass] International Phonetic Association. Full ipa chart. `https://www.internationalphoneticassociation.org/content/full-ipa-chart`. Online; accessed 15 June 2025.

[AWS] AWS. Boto3 documentation. `https://boto3.amazonaws.com/v1/documentation/api/latest/index.html`. Online; accessed 14 June 2025.

[CBC^{+}21] Alexis Conneau, Alexei Baevski, Ronan Collobert, Abdelrahman Mohamed, and Michael Auli. Unsupervised cross-lingual representation learnig for speech recognition. *Interspeech 2021, 30 August – 3 September, 2021, Brno, Czechia*, 2021.

[CC] CUNY-CL. Wikipron. `https://github.com/CUNY-CL/wikipron`. Online, accessed 7 May.

[CMU] CMUSphinx. g2p-seq2seq. `https://github.com/cmusphinx/g2p-seq2seq`. Online; accessed 15 April 2025.

[Doc] Docker Inc. Docker Docs. `https://docs.docker.com/`. Online; accessed 14 June 2025.

[Goo] Google. Material Design 3. `https://m3.material.io/`. Online; accessed 12 June 2025.

[GP23] Carol-Luca Gasan and Vasile Păiș. Investigation of romanian speech recognition improvement by incorporating italian speech data. *Linguistic Resources and Tools for Natural Language Processing, Proceedings of the 18th International Conference, Brașov, 11-14 December 2023*, 2023.

[Huga] HuggingFace. Datasets. `https://huggingface.co/docs/datasets/en/index`. Online; accessed 8 June 2025.

[Hugb] HuggingFace. Inference Endpoints. `https://huggingface.co/inference-endpoints/dedicated`. Online; accessed 9 June 2025.

[Hugc] HuggingFace. PEFT. `https://huggingface.co/docs/peft/en/index`. Online; accessed 8 June 2025.

[Hugd] HuggingFace. Transformers. `https://huggingface.co/docs/hub/en/transformers`. Online; accessed 8 June 2025.

[OKH] OKHttp. Overview-OkHttp. `https://square.github.io/okhttp/`. Online; accessed 12 June 2025.

[Pao16] Sandra Paoli. A short guide to italian phonetics and phonology for students of italian paper v. *Oxford Univeristy, Faculty of Linguistics, Philology and Phonetics*, 2016.

[PBMI$^+$24] Vasile Păiș, Verginica Barbu Mititelu, Elena Irimia, Radu Ion, and Dan Tufiș. Under-represented speech dataset from open data: Case study on the romanian language. *Appl. Sci. 2024, 14, 9043*, 2024.

[RACa] RACAI. Korap. `https://korap.racai.ro/`. Online; accessed 8 June 2025.

[RACb] RACAI. Nlp-cqp. `http://89.38.230.23:1234/`. Online; accessed 8 June 2025.

[RACc] RACAI. Ocqp. `http://89.38.230.23/corola_sound_search/index.php`. Online; accessed 8 June 2025.

[Ram] Sebastian Ramirez. FastAPI. `https://github.com/fastapi/fastapi`. Online; accessed 14 June 2025.

[Rob] James Robert. pydub. `https://pypi.org/project/pydub/`. Online; accessed 14 June 2025.

[RWKX+22] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. *Proceedings of the 40th International Conference on Machine Learning*, 2022.

[Sal10] Claudia S. Salcedo. The phonological system of spanish. *Rioja University, Revista de Lingüística y Lenguas Aplicadas*, 2010.

[SBCA19] Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. wav2vec: Unsupervised Pre-training for Speech Recognition. *arXiv*, 2019.

[spr] spring-media. DeepPhonemizer. `https://github.com/spring-media/DeepPhonemizer`. Online, accessed 7 May.

[TBMI+19] Dan Tufiș, Verginica Barbu Mitelu, Elena Irimia, Vasile Păiș, Radu Ion, Nils Diewald, Maria Mitrofan, and Mihaela Onofrei. Little strokes fell great oaks. creating corola, the reference corpus of contemporary romanian language (corola). *RRL, vol. LXIV, 3, p. 227-240*, 2019.

[Tea] FFmpeg Team. About FFmpeg. `https://ffmpeg.org/about.html`. Online; accessed 14 June 2025.

[Tec] Technical University of Iași. Voiced sounds of the romanian language. `http://www.etc.tuiasi.ro/sibm/romanian_spoken_language/en/structura_fonologica.htm`. Online; accessed 15 April 2025.

[TSV+08] G. Tur, A. Stolcke, L. Voss, J. Dowding, B. Favre, R. Fernandez, M. Frampton, M. Frandsen, C. Frederickson, M. Graciarena, D. Hakkani-Tur, D. Kintzing, K. Leveque, S. Mason, J. Niekrasz, S. Peters, M. Purver, K. Riedhammer, E. Shriberg, J. Tien, D. Vergyri, and F. Yang. The CALO meeting speech recognition and understanding system. *2008 IEEE Spoken Language Technology Workshop*, 2008.

[Whi] Whisper. The Whisper Models. `https://whishper.net/reference/models/`. Online; accessed 8 June 2025.

[Wik] Wikipron and victors3136. Training data. `https://github.com/victors3136/Training-Data/tree/main`. Online; accessed 12 May 2025.

[Zga19] Andrej Zgank. Cross-lingual speech recognition between languages from the same language family. *Proceedings of the Romanian Academy, Series A, Volume 20, Number 2/2019, pp. 189–196*, 2019.