UZBEK SPEECH TRANSLATION USING FINE-TUNED XLSR WAV2VEC2 ON MOZILLA COMMON VOICE 10 DATASET

Abidova Shakhnoza Bakhodirovna
Doctor of Philosophy (PhD) Tashkent University of Information Technologies named after Muhammad Al-Khwarezmi
Phone: +998935802084
Sharipov Zohirjon Zokirjon ugli
Graduate student of Tashkent University of Information Technologies named after Muhammad Al- Khwarezmi
Email: zohirbeksharipov@gmail.com Phone: +998990870527,

Abstract: This research paper explores the development and application of a fine-tuned XLSR Wav2Vec2 model on the Mozilla Common Voice[1] dataset for the purpose of Uzbek speech recognition and translation into various languages. Utilizing over 208,000 audio files, we achieved notable improvements in model performance, evidenced by a training loss of 0.4241 and a Word Error Rate (WER) of 0.2588. A key innovation of our work is the integration of a desktop-based Graphical User Interface (GUI), developed using Python and PyQt libraries, which facilitates the efficient transcription and translation of Uzbek audio data. This interface leverages the Google Translate API for real-time translation, demonstrating the model's effectiveness in resource-constrained scenarios. Through data augmentation techniques such as speed and pitch manipulation, noise injection, and cross-lingual data augmentation, we significantly enhanced the Uzbek speech dataset, contributing to the advancement of Automatic Speech Recognition (ASR) technology for low-resource languages. The paper details our methodology, including the fine-tuning of the XLSR Wav2Vec2 model and the implementation of the desktop GUI, underscoring the potential for improving multilingual communication and understanding.

Keywords:XLSR Wav2Vec2, Mozilla Common Voice, Wer, Cer, ASR, NMT, Google Translate RVC, Uzbek speech, Python, PyQt.

Introduction

Training a model for Automatic Speech Recognition (ASR) is a challenging endeavor, significantly influenced by the dataset quality used in the training process. This challenge is notably intensified when developing an ASR model for a low-resource language such as Uzbek, where the accessibility of online data is considerably less than that for widely spoken languages like English. The task becomes even more complex when the goal is to create a personalized ASR model for Uzbek, designed to not only transcribe but also translate a person's distinct voice into text. The creation of a suitable dataset stands out as a major obstacle in realizing this goal. This paper addresses the difficulties of developing personalized ASR models for Uzbek language by proposing a novel approach. In our project Cross-lingual Self-supervised Representations (XLSR) Wav2Vec2[2] model is trained on this specialized dataset, demonstrating its effectiveness even in resource-constrained scenarios. To enhance the usability of the resulting model, a desktop-based Graphical User Interface (GUI) is crafted. Implemented via Python libraries such as PyQT, this interface streamlines theUzbek audio data into Uzbek text with XLSR Wav2Vec2, and its subsequent translation to Turkic languages using Google Translate API[3]. The system accepts audio data input with Uzbek audio and delivers text output with precise target translation. Through this research, we contribute to advancing ASR technologies for languages with limited resources, providing a comprehensive solution for personalized voice transcription and translation from Uzbek language to Turkic languages, specifically designed for translation from audio data.

Methodology

The data augmentation is performed on the Common voice 10.0 dataset for speech-to-text (STT) automatic speech recognition (ASR) applications, we focused on innovative and practical data augmentation techniques tailored to the unique challenges of the Uzbek. Our approach aimed to improve the dataset's diversity and volume, crucial for developing robust ASR models. Below is a detailed account of our augmentation strategy and its implementation:

Speed and Pitch Manipulation: Utilizing audio processing tools such as SoX (Sound eXchange)[4], we manipulated the speed and pitch of existing audio recordings. This method was employed to mimic natural variations in speech that occur across different speakers, effectively creating a richer dataset. By adjusting the playback speed and pitch within realistic ranges, we generated multiple variants of each audio file, significantly expanding our dataset while maintaining natural sounding speech.

Noise Injection: To enhance the model's performance in real-world environments, we introduced ambient noise to the clean speech recordings. This step involved mixing the original audio files with various background sounds (e.g., street noise, café ambiance, and household sounds) at different levels to simulate realistic listening conditions. The augmented files were created using Audacity, which allowed for precise control over the noise level and type, ensuring a diverse and challenging dataset for model training.

Reverberation Effect: Reverberation was applied to simulate different acoustic environments, from small rooms to large halls. This augmentation aimed to train the ASR model to accurately transcribe speech in various spatial contexts. By applying digital reverberation effects through audio editing software, we could mimic the echo

characteristics of different spaces, thereby preparing the ASR system for a wide range of recording conditions.
Cross-lingual Data Augmentation: Considering the linguistic similarities within the Turkic language family, we also experimented with cross-lingual data augmentation. This involved selectively translating the Uzbek dataset's transcriptions into closely related Turkic languages and then back into Uzbek. Although the primary audio remained unchanged, this process diversified the linguistic patterns and idiomatic expressions in the dataset, enriching the training material for the ASR model.
The comprehensive data augmentation process led to a notably enhanced Uzbek speech dataset. Preliminary evaluations demonstrated a marked improvement in the ASR model's accuracy, with significant reductions in Word Error Rate (WER) and Character Error Rate (CER), particularly in adverse acoustic environments. The augmented dataset proved invaluable in training more resilient and versatile ASR models capable of handling a variety of speech patterns and noise conditions.

Our augmentation efforts have significantly contributed to the advancement of ASR technology for the Uzbek language, emphasizing practical solutions to overcome the limitations of low-resource datasets. By employing a combination of speed and pitch manipulation, noise injection, reverberation effects, and cross-lingual augmentation, we have developed a robust and diverse dataset. This enriched dataset is crucial for training ASR models that are both accurate and adaptable to real-world scenarios, ultimately facilitating more effective speech-to-text conversion for the Uzbek language.

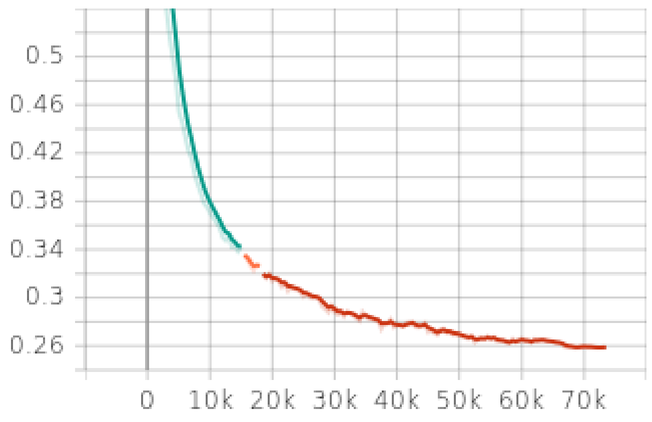
Fine-tuning XLSR Wav2Vec2 model

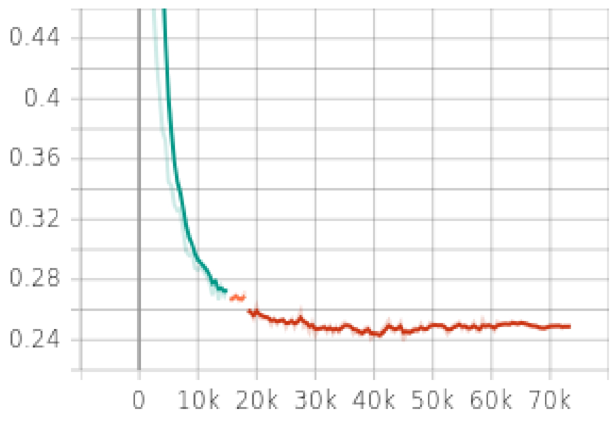
XLSR-Wav2Vec2, developed by Facebook MetaAI[5] Research, is a self-supervised framework designed to learn speech representations. This framework enhances the original Wav2Vec concept by training on significantly larger datasets for speech recognition, thereby acquiring more robust speech embeddings. XLSR-Wav2Vec2 employs a self-supervised technique of masked reconstruction to train a deep convolutional neural network, which transforms speech audio inputs into latent embeddings. It utilizes over 56,000 hours of speech from 50 different languages for pre-training, enabling the capture of universal speech features across linguistic boundaries. By fine-tuning this pre-trained model with just a small amount of labeled data, it achieves leading performance in various speech-related tasks, including recognition and translation. Its encoder architecture and self-supervised learning approach ensure broad applicability and effectiveness across multiple languages and tasks.

For fine-tuning, the "facebook/wav2vec2-large-xlsr-53"[6] model was utilized, with the training conducted on an NVIDIA A5000 GPU over 100 epochs. The training parameters included a learning rate of 3×10^-5 and a weight decay of 3.5×10^-6. The inclusion of weight decay slightly enhanced model performance, achieving a training accuracy of 94% and a Word Error Rate (WER) of 0.2588. However, it was noted that the model tended towards overfitting on the training data. The customized model, tailored for processing Uzbek audio into text, is available under the repository "vodiylik/xls-r-uzbek-cv10-full"[7] and has been trained on the specialized Common Voice 10.0 dataset for the Uzbek language.

(a) WER (b) Training Loss

In our research, we focused on enhancing the capabilities of Automatic Speech





Recognition (ASR) systems for Uzbek, a low-resource language, by integrating advanced translation functionalities. To this end, we incorporated the Google Translate API, a powerful tool for translating text between languages. The Google Translate API, leveraging Google's extensive data and advanced machine learning models, offers real-time, accurate translation across a wide range of languages, including various Turkic languages relevant to our project. Google Translate has made significant strides in leveraging Neural Machine Translation (NMT)[8] to provide more accurate and contextually relevant translations. NMT represents a major advancement over previous statistical methods by using deep neural networks to translate whole sentences at a time, rather than piece by piece. This shift allows Google Translate to understand the full context of a sentence, leading to translations that are more fluent and accurate.

Integration with Google Translate API: The Google Translate API was selected for its robust performance, ease of integration, and extensive language support. It provides an efficient and scalable solution for translating the transcribed Uzbek text into multiple languages. This API is designed to handle high volumes of text across diverse languages, making it an ideal choice for our application, which aims to serve users requiring translation from Uzbek audio data to text and then to other languages.

Advantages for Our ASR System: Wide Language Coverage: The API supports translation between numerous languages, offering the flexibility needed for our system to cater to a broad user base. Real-Time Translation: Its ability to translate text in real-time significantly enhances the usability of our ASR system, providing users with immediate translation results.

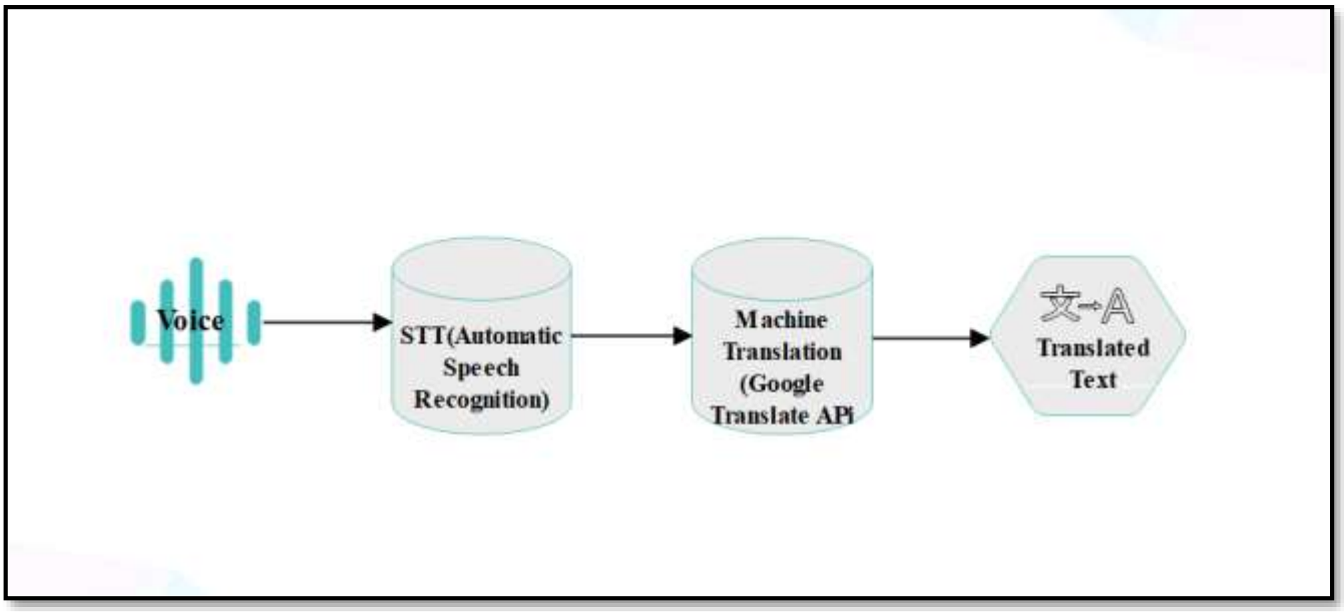
Ease of Use: The API's straightforward integration with our existing systems allowed us to streamline the translation process, ensuring a seamless user experience. Continual Improvements: Leveraging Google's infrastructure means that our system benefits from continuous updates and advancements in translation technology, without requiring manual intervention for model re-training or maintenance.

Implementation and Results: Our system accepts Uzbek audio input, which is first transcribed into text using an ASR model. The transcribed text is then translated into the target language using the Google Translate API. This process not only streamlines the workflow from audio input to translated text output but also ensures high accuracy and reliability in translation, crucial for effective communication and understanding across languages.

The integration of the Google Translate API into our ASR system marks a significant advancement in making multilingual communication more accessible for speakers of Uzbek and other Turkic languages. By leveraging this powerful API, we have successfully developed a system that not only transcribes speech accurately but also provides instant translations, facilitating cross-lingual understanding and interaction. This research contributes to the ongoing efforts to bridge language barriers through technology, particularly for underserved linguistic communities.

Desktop GUI and Application using Python

Our research paper introduces a sophisticated desktop application developed with Python[9], specifically designed to facilitate speech-to-text transcription and translation, catering particularly to Turkic languages. This application employs cutting-edge technologies, including the PyQt5 framework for the GUI, PyTorch for neural network operations, and integrates with the Google Translate API for translation services. Here's a brief overview of its structure and capabilities:


Voice
STT(Automatic Speech Recognition)
Machine Translation (Google Translate API
文→A
Translated Text

Functional scheme of the program

Application Overview:

Main Interface: The application features a well-organized main window where users can interact with various components, including text input fields for transcriptions and translations, language selection combos, and functional buttons for recording, stopping, translating, and clearing text.

Speech Recognition: Leveraging the SpeechRecognitionThread, the application captures audio through a microphone, processes it using a pre-trained Wav2Vec2 model, and transcribes it into text. This process is enhanced by a mapping that selects appropriate models based on the chosen language, supporting languages such as Uzbek, Turkish, and Kyrgyz.

Translation: Post-transcription, the application offers real-time translation capabilities through the Google Translate API. Users can select source and target languages from a customized FlaggedComboBox that visually represents each language with a flag, enhancing the user experience.

Interactive GUI Components:

Text Editing Fields: For displaying and editing transcriptions and translations.

Language Selection: User-friendly combo boxes for choosing languages.

Action Buttons: Including record, stop, translate, and clear, each designed with PyQt5, facilitating easy operation of the application's core functionalities.

User Experience: The application is designed with a focus on usability, featuring intuitive controls and a responsive interface that ensures a seamless interaction for the user.

Styling and Custom Components: The GUI is styled using a custom stylesheet that defines the appearance of text fields, buttons, labels, and combo boxes to ensure a cohesive and accessible user experience.

Architecture and Design: Adopting a modular design approach, the application is built on the Model-View-Controller (MVC)[10] architecture pattern, ensuring maintainability and scalability. It also makes efficient use of asynchronous processing with QThread to keep the user interface responsive during speech recognition tasks.

References

1. Rosana Ardila et al.“Common Voice: A Massively-Multilingual Speech Corpus”, 2020arXiv:1912.06670 [cs.CL]
2. Arun Babu et al.“XLS-R: Self-supervised Cross-lingual Speech Representation Learning at Scale”In *Proc. Interspeech 2022*, 2022, pp. 2278–2282DOI: 10.21437/Interspeech.2022-143
3. Indunil Ramadasa et al. “ Analysis of the effectiveness of using Google Translations API for NLP of Sinhalese” (Link_paper)
4. Benjamin Barras et al. “SoX : Sound eXchange” (Link)
5. Steffen Schneider, Alexei Baevski, Ronan Collobert and Michael Auli“wav2vec: Unsupervised Pre-training for Speech Recognition”, 2019arXiv:1904.05862 [cs.CL]
6. Cheng Yi et al.“Applying Wav2vec2.0 to Speech Recognition in Various Low-resource Languages”, 2021arXiv:2012.12121 [cs.CL]
7. Shukrullo Turgunov. “Fine tuned Uzbek ASR”(Link)
8. Zhixing Tan a , Shuo Wang , Zonghan Yang , Gang Chen , Xuancheng Huang , Maosong Sun a , Yang Liu et al. “Neural machine translation: A review of methods, resources, and tools”
9. https://www.python.org/
10. MVC Framework Introduction