Punjabi Speech to Text and Keyword Searching

CP-303: B.Tech Project-II

Submitted by:

Akarshi Roy Choudhury (2021EEB1149)

Ashwini Sahane (2021EEB1159)

Bhumika Chaudhari (2021EEB1160)

Lokesh Jassal (2021EEB1185)

Under the guidance of

Dr. J.S. Sahambi

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY ROPAR

2025

Acknowledgement

We would like to extend our deepest gratitude to Dr. J.S. Sahambi, who provided invaluable guidance and support throughout our CP303 (BTP) project this semester . His expertise and encouragement have been instrumental in shaping the outcomes of this effort.

We are sincerely thankful to Dr. Somdev Kar who provided the resources and helped us contact the Linguistic Data Consortium for Indian Languages (LDCIL), Mysore from where we received the authentic dataset for our project.

We are also grateful to the Teaching Assistants (TAs): Mr. Inderjeet and Mr. Aditya Bharadwaj for their constructive feedback.

Finally, we would like to acknowledge LDCIL, Mysore for providing us the desired and required dataset, which was crucial for this project.

Content

1. Objective

- Punjabi Speech to Text Conversion
- Fine-tuning the Models
- Creating and Saving a .txt file for each predicted transcript
- Exact Keyword Search
- Related Keyword Search
- Semantic Search
- Basic GUI

Dataset Used: LDCIL Punjabi Raw Speech Corpus

Use Case: Punjab Police

2. Dataset

We have used the LDCIL Punjabi Raw Speech Corpus for training the Punjabi speech to text models. We have used the Sentences part of the dataset in accordance to our use case. The details of the sentence dataset are as follows:

- Total: 11168 sentences, 8:58:34 duration
- Male: 5577 sentences, 04:28:10 duration
- Female: 5591 sentences, 04:30:24 duration
- Regions: Malwai, Puadhi, Doabi

The age group wise details have been tabulated below:

Gender	Male			Female		
Age Group	16-20 years	21-50 years	51+ years	16-20 years	21-50 years	51+ years
Sentences Spoken	550	3353	1674	673	3393	1625
Duration	00:25:45	02:36:53	01:25:32	00:30:22	02:34:53	01:25:09

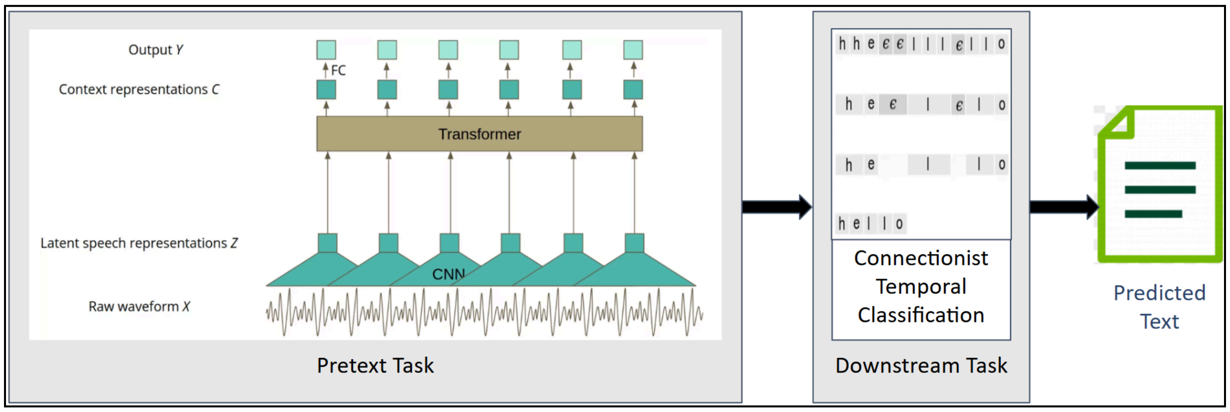
To make the models learn better about sentence completion and pauses, we have concatenated 4 audios into one, resulting in 11168/4 = 2792 audio files. Further, we have randomly divided them into training and testing datasets (80-20 split) of 2234 and 558 audio files respectively.

3. Punjabi Speech to Text Conversion

3.1 WAV2VEC2

The Wav2Vec model converts raw audio into meaningful speech representations. The model consists of a feature encoder and a context network. The feature encoder, a 7-layer CNN, processes 16kHz audio into latent vectors by capturing local audio patterns and reducing dimensionality. These vectors are passed to the context network (a transformer), which models long-range dependencies to produce context-rich representations. For fine-tuning on downstream tasks like ASR, the model uses Connectionist Temporal Classification (CTC), which aligns the context vectors with target transcriptions. CTC introduces a blank token for handling pauses, allows repeated predictions, and decodes the final text by collapsing repeats and removing blanks using greedy or beam search decoding.

The block diagram for Wav2Vec for ASR has been shown below:



Reference: https://neurosys.com/blog/wav2vec-2-0-framework

The conventional Wav2Vec model works well for English language, however, for Punjabi audio input, it gives English text output, that too, not much accurate. So, we found the “manandey/wav2vec2-large-xlsr-punjabi” model on Hugging Face, which has been trained using the Mozilla Common Voice dataset for Punjabi speech to text conversion.

2.1.1 RESULTS FOR “manandey/wav2vec2-large-xlsr-punjabi” MODEL

We have used Hugging Face's trainer API in order to train the model. We have used the LDCIL test dataset for testing the original as well as the fine-tuned model. Word Error Rate (WER) has been used as the evaluation metric. WER measures the difference between the model's predictions and actual transcriptions.

$WER = (Substitutions + Insertions + Deletions)/Total\,Words\,in\,Reference$

The results obtained are as follows:

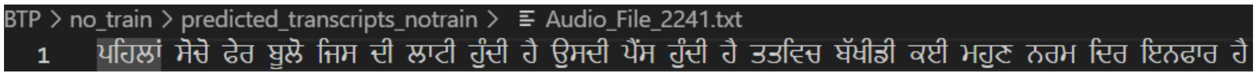
Without training: WER = 0.78

With training: WER = 0.7711

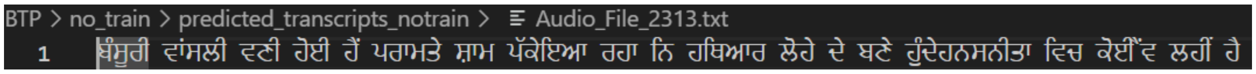
Training only the last layer: 0.99

The snapshots of some of the predicted transcripts are as shown below:

- Predicted Transcripts on the original model (without training):

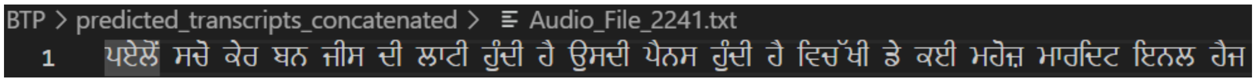


```
BTP > no_train > predicted_transcripts_notrain > Audio_File_2241.txt
ਪਹਿਲਾਂ ਸੋਚੋ ਫੇਰ ਬੁਲੋ ਜਿਸ ਦੀ ਲਾਟੀ ਹੁੰਦੀ ਹੈ ਉਸਦੀ ਪੈਂਸ ਹੁੰਦੀ ਹੈ ਤਤਵਿਚ ਬੱਖੀਡੀ ਕਈ ਮਹੁਣ ਨਰਮ ਦਿਰ ਇਨਫਾਰ ਹੈ
```

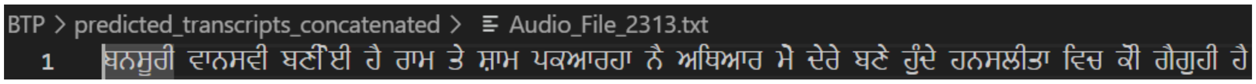


```
BTP > no_train > predicted_transcripts_notrain > Audio_File_2313.txt
ਬੰਸੁਰੀ ਵਾਂਸਲੀ ਵਣੀ ਹੋਈ ਹੈਂ ਪਰਾਮਤੇ ਸ਼ਾਮ ਪੱਕੇਇਆ ਰਹਾ ਨਿ ਹਥਿਆਰ ਲੋਹੇ ਦੇ ਬਣੇ ਹੁੰਦੇਹਨਸਨੀਤਾ ਵਿਚ ਕੋਈਂਵ ਲਹੀਂ ਹੈ
```

- Predicted Transcripts on training the entire model:

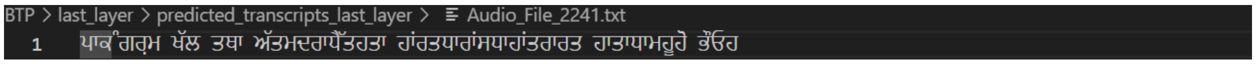


```
BTP > predicted_transcripts_concatenated > Audio_File_2241.txt
ਪਏਲੋਂ ਸਚੋ ਕੇਰ ਬਨ ਜੀਸ ਦੀ ਲਾਟੀ ਹੁੰਦੀ ਹੈ ਉਸਦੀ ਪੈਨਸ ਹੁੰਦੀ ਹੈ ਵਿਚੱਖੀ ਡੇ ਕਈ ਮਹੋਜ਼ ਮਾਰਦਿਟ ਇਨਲ ਹੈਜ
```

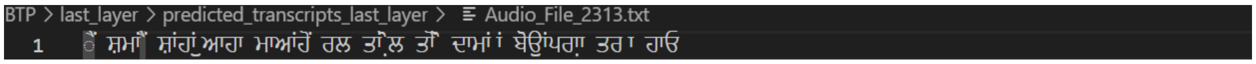


```
BTP > predicted_transcripts_concatenated > Audio_File_2313.txt
ਬਨਸੁਰੀ ਵਾਨਸਵੀ ਬਣੀਂਈ ਹੈ ਰਾਮ ਤੇ ਸ਼ਾਮ ਪਕਆਰਹਾ ਨੈ ਅਥਿਆਰ ਮੇ ਦੇਰੇ ਬਣੇ ਹੁੰਦੇ ਹਨਸਲੀਤਾ ਵਿਚ ਕੌਂ ਗੈਗੁਹੀ ਹੈ
```

- Predicted Transcripts on training only the last layer:



```
BTP > last_layer > predicted_transcripts_last_layer > Audio_File_2241.txt
ਪਾਕਂਗਰ਼ਮ ਖੱਲ ਤਥਾ ਅੱਤਮਦਰਾਪੈੱਤਰਹਤਾ ਹਾਂਰਤਧਾਰਾਂਸਧਾਹਾਂਤਰਾਰਤ ਹਾਤਾਧਾਮਹੂਹੋ ਭੌਓਹ
```



```
BTP > last_layer > predicted_transcripts_last_layer > Audio_File_2313.txt
ੈਂ ਸ਼ਮਾੈਂ ਸ਼ਾਂਹਾਂ੍ਆਹਾ ਮਾਆਂਹੋਂ ਰਲ ਤਾੌ਼ਲ ਤੌਂ ਦਾਮਾਂ ਂ ਬੋਉਂਪਰਗ਼ਾ ਤਰ ਾ ਹਾਓ
```

Since, the Wav2Vec2 model didn't seem to be giving good results on the Punjabi dataset, hence we moved on to trying Whisper AI.

3.2 WHISPER AI

Whisper AI is an advanced speech recognition model developed by OpenAI, designed to convert spoken language into written text with high accuracy. Trained on a vast and diverse dataset of multilingual and multitask audio, Whisper is capable of understanding and transcribing a wide range of languages and accents. It supports applications like automated transcription, translation, and voice-controlled interfaces, making it a versatile tool for developers and researchers. The model has been for English Speech-to-text conversion for over 6,80,000 hours of multilingual and multitask (transcription, translation and identification of the language used) of which approximately 437,000 hours of this dataset consist of English audio, with the remaining 243,000 hours spanning multiple languages.

3.2.1 WORKING OF THE BASIC MODEL

This AI works on the transformer architecture. The input data is an audio file (can be of any form: .mp3, .wav etc, all types are supported) followed by encoding (processing of the data, initialization of tokens in the transformer block) and decoding (outputs the corresponding transcript). Explained thoroughly further -

Encoding:

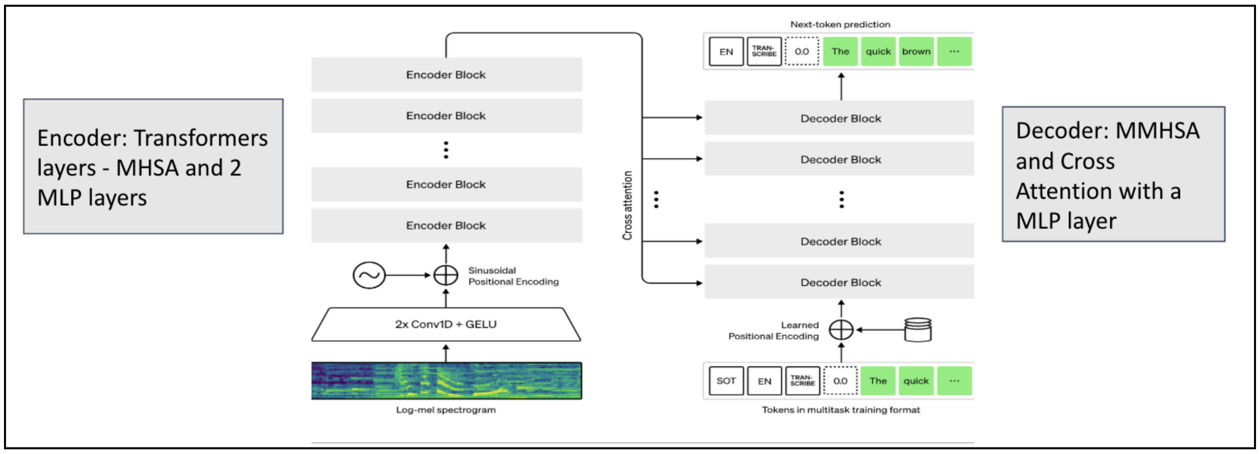
In an overall manner, it first tries encoding the input audio signal, converts it into the frequency domain and accordingly plot a spectrogram w.r.t the same on a log-mel scale (includes logarithmic transformation of the frequencies of the audio and also help identify the distance between different pitches). This audio is further processed via Multi-Head Self-Attention (MHSA) and Multi Layer Perceptron (MLP). In simpler words, there are multiple transformer layers and at each layer, MHSA captures dependencies across the time-frequency dimensions of the audio signal and also helps the model focus on important features like phonemes, noise patterns, or tonal changes. MLP also known as the feedforward block comprises two layers in itself. The first layer expands the dimensionality for a high number of features thereby providing better training. The second layer compresses it back to ensure compatibility with the rest of the model. Besides, MLP also includes

Non-Linear Transformation (ReLU and GeLU), since these linear layers discussed above can't capture the non-linearity in the data (if any), thus it needs non-linearity for better functionality.

Decoding:

The encoding layers initialize a token w.r.t the audio processed, there's a start-of-sequence (SOS) token or some previously produced tokens. This layer includes Masked Multi-Head Self-Attention (here, it ensures that the model only attends to previously generated tokens during training or inference), Cross Attention (Links the decoder with the encoder by attending to the encoded audio representation/tokens). Next up comes MLP, here there's backpropagation involved, (whilst the multiple layers, the error at the output is decreased by back propagating through all those layers till it reduces or goes less than the threshold that we set).

A softmax layer predicts the next token in the sequence, selecting from a vocabulary of text tokens. Accordingly (based on the tokens), the transcript is now generated at the output layer. The block diagram below depicts the process discussed above:

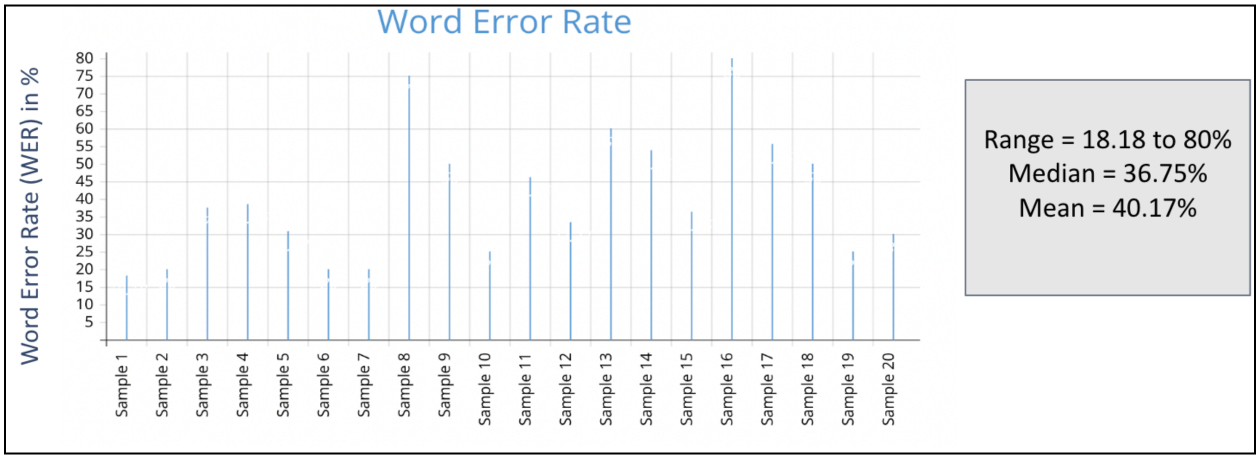


Reference: OPEN AI WEBSITE (https://openai.com/index/whisper/)

3.2.2 FINE-TUNING THE MODEL:

Since Whisper AI was known for good enough results in English language, for Punjabi audio inputs, it gave a Punjabi text output however in English transcript. Thus there was a need to fine tune/ train the existing whisper AI model with audio files and its corresponding transcripts in Punjabi for our desired result. For the same, we found the "DrishtiSharma/whisper-large-v2-punjabi-700-steps" model on Hugging Face. It is a fine-tuned version of OpenAI's Whisper Large-v2, optimized for automatic speech recognition (ASR) in Punjabi. This model was trained using the Mozilla Common Voice 11.0 dataset, specifically focusing on the Punjabi language (pa-IN configuration).

The model uses ADAM optimizer (different learning rates for different parameters) with over 700 steps of optimization. With a loss at completion (w.r.t loss function, a performance metric) of nearly 22%, the model gave a Word Error Rate (WER) of nearly 25%. We tested this model on 20 audio clips and the WERs ranged from 18.18 to 80% with a median of 36.75% and Mean of 40.17%. We therefore thought of training the model further for greater accuracy.



The model used performs speech-to-text transcription and evaluation for a dataset of audio files. It begins by defining dataset paths and loading a reference transcription file (test.tsv) containing audio file names and their corresponding text transcriptions. The script processes the first 20 entries, updating file paths to include their full locations and .wav extensions. It loads a pre-trained Whisper model and processor from Hugging Face for Punjabi transcription ("DrishtiSharma/whisper-large-v2-punjabi-700-steps"). Using this setup, the script iterates through each audio file, checks its existence, and processes it using librosa for feature extraction. The model generates transcriptions, which are stored alongside the reference text. The Word Error Rate (WER) metric is calculated for each transcription using the jiwer library, quantifying the difference between the predicted and reference transcriptions. Finally, the WER scores are saved to a text file (wer_scores.txt) for review, enabling evaluation of the model's transcription performance.

3.2.3 TRAINING "DrishtiSharma/whisper-large-v2-punjabi-700-steps" MODEL

The testing data that we used had small clips of nearly 10-15s duration of Punjabi audio. We used 100 such samples for training initially and 20 more testing samples (same as the ones used above) for test purposes. We loaded the Whisper model (WhisperForConditionalGeneration) and its processor (WhisperProcessor) from our pre-trained model from Hugging Face. The WhisperProcessor is responsible for preprocessing audio and text into formats that the Whisper model can handle. We then tried preprocessing the audio clips to convert audio clips into feature tensors suitable for Whisper. It ensures that audio is processed with the correct sampling rate and padding (up to 3000 frames). It also tokenizes the transcript texts to prepare labels for training. Next up, we collated the data i.e. added dynamic padding to ensure the input tensors are appropriately padded to a common size.

With the following training arguments, we tried training our model:

1) **output_dir:** for saving the model.
2) **Evaluation_strategy**: set to evaluate the model at each epoch.
3) **learning_rate**, **batch_size**, **num_train_epochs** etc., for training configuration.
4) **gradient_accumulation_steps**: This is set to 64, which helps when using a smaller batch size but still allows accumulating gradients over multiple steps before updating the model parameters, improving memory efficiency.

5) **fp16=True** enables mixed-precision training to speed up computation and reduce memory usage.
6) **optim="adamw_torch"**: Specifies the use of the memory-efficient AdamW optimizer, which is commonly used in large-scale training.

The code is framed such that one has to only upload the audio file (in any extension say .mp3, .wav etc) and they'll get its corresponding transcript directly with an estimation of less WER than previously available pretrained model.

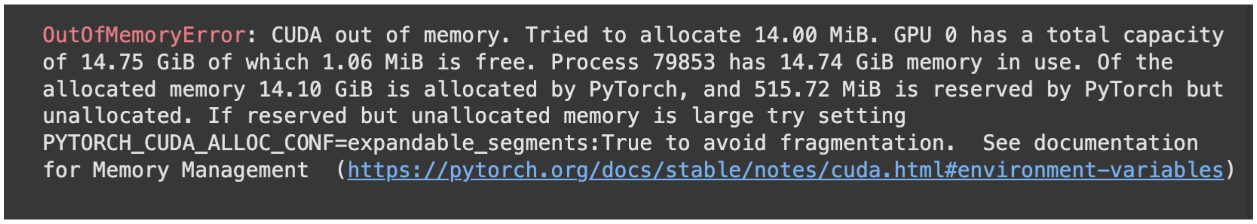
The preprocess_data function processes individual examples from the dataset. It reads the path to an audio file, checks if the file exists, and loads it using torchaudio. The audio data is converted from a tensor to a NumPy array for further processing. If loading fails (e.g., file missing or corrupted), the error is logged, and the function returns None. For valid audio files, the function outputs a dictionary containing the audio data and its corresponding transcript.

After defining the preprocess_data function, it is applied to the dataset using the data.map method, which processes each example and removes unnecessary columns like audio_file and transcript. Examples that returned None are filtered out, and the remaining data is split into training and testing sets.

The preprocess_function converts the audio data into input features using the Whisper processor's feature extractor. It applies padding to ensure all audio inputs are of the same length (3000 frames). The text labels (transcripts) are tokenized using the Whisper tokenizer, also with padding and truncation. Both inputs and labels are converted to PyTorch tensors.

The collate_fn function dynamically pads the input features and labels during training to ensure consistent batch sizes. It stacks the tensors for audio input and uses pad_sequence to align label lengths with a padding token. This function is essential for handling variable-length data during model training. Together, these preprocessing steps prepare the dataset for training a Whisper model. This dataset is later trained on the Hugging Face model via the arguments mentioned above.

3.2.4 LIMITATION:



```
OutOfMemoryError: CUDA out of memory. Tried to allocate 14.00 MiB. GPU 0 has a total capacity
of 14.75 GiB of which 1.06 MiB is free. Process 79853 has 14.74 GiB memory in use. Of the
allocated memory 14.10 GiB is allocated by PyTorch, and 515.72 MiB is reserved by PyTorch but
unallocated. If reserved but unallocated memory is large try setting
PYTORCH_CUDA_ALLOC_CONF=expandable_segments:True to avoid fragmentation.  See documentation
for Memory Management  (https://pytorch.org/docs/stable/notes/cuda.html#environment-variables)
```

Whisper is a large Model (includes dimensionality expansion for precision on transformer level) and thus requires a good enough GPU to train it. Irrespective of decreasing the batch sizes and training data, increasing the gradient accumulation, importing python libraries like accelerate etc and by deciding an optimal learning rate (all for memory optimization), there were no improvements. Even after training it on GPU, we still got this memory error since the GPU allotted didn't hold significant memory to train all 15 Billion parameters of Whisper AI. The entire model got trained on GPU only when we gave only 1 input video for training. One had to iteratively train 2233 times for the 2233 concatenated clips that we had by saving & re-training the model which could have been quite tedious. Thus, we thought of applying transfer learning, particularly freezing the encoder and training only the decoder along with the last Fully Connected (FC) Layer. It has been elaborated more in the following section.

3.2.5 FURTHER FINE-TUNING AND TRAINING ON LDCIL DATASET

The high Word Error Rate (WER) even for the available fine tuned w.r.t. Punjabi Speech and corresponding transcripts of the Drishti Sharma model on Hugging Face made us train the available model further with the LDCIL Dataset that we got, as mentioned previously. Since we were unable to train the available model as it's extensive and holds around 15B parameters, unless we have a very good GPU. The encoder part of the whisper model is the extensive part which beholds the maximum number of parameters. The decoder being relatively way smaller than the encoder allows it to be trainable enough on the available GPU. The decoder along with the last FC layer hold approximately 60M parameters. The model works in the following manner:

1) We first load a .txt file mapping the audio filename to the corresponding audio files. It then creates a Hugging face dataset object.
2) We then load the .wav file and ensure that it is mono channeled and not stereo. We then resample these audio files to an uniform frequency of 16 kHz - as per the specs in the Whisper AI model. This way, we first load the audio waveform.
3) We then remove the entries where Audio processing has failed and these files are returned as none.
4) We then load Drishti Sharma's Hugging face Punjabi Speech model, both the tokenizer and feature extractor. The steps till here are the same for training both, the entire model and part of it (here, decoder).
5) Next, we now freeze all the parameters except for the decoder and the last FC layer. This layer (the FC layer) maps internal representations to vocabulary logits.
6) We then convert this waveform obtained to log-mel scale followed by dynamic labelling to 3000 frames (~30s) the audio since that's how whisper has been trained. It pads the input sequence with less than 3000 frames and truncates the one more than that thus creating an issue for larger audio files. To combat this issue, we first break the audio files into chunks of 30s each, process the transcript for each of these chunks and then concatenate it to give the final result for the input audio. We then tokenize (create labels) these transcripts and then return these tokens/labels in a form of tensors
7) The collate function load the input features and the corresponding transcript labels/ tokens.
8) We have used the following training arguments to train this model:
    - batch_size = 1 (more batches require more memory thus kept as 1 for memory optimization)
    - Epoch = 1(The model didn't run for more than 1 epoch, it required more memory and run time for training on more epochs in case of training on the entire dataset provided, all the 2233 concatenated clips
    - gradient_accumulation_steps=64 → simulates large batch size.
    - gradient_checkpointing=True → saves memory.
    - fp16=True → uses half-precision for speed/memory.
    - logging_steps=1 → frequent logs
    - save_strategy="epoch" → saves once per epoch.

9) Since the model wasn't being trained for more than one epoch, for better results and even better & refined weights, we saved it after every epoch and trained it all over again iteratively. For the same, we modified trainer.train( ) such that it resumes training from the latest checkpoint.
10) We got around **66387200** trainable parameters which boil down to around 60M trainable parameters of the decoder + last FC layer.

3.2.6 RESULTS OF THE FINE-TUNED MODEL

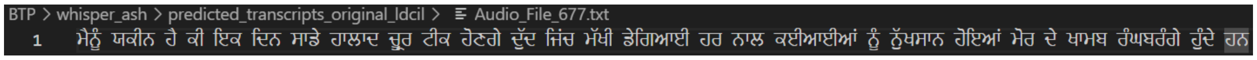
The results obtained are as follows (for 100 samples):

Without training: WER = 0.625

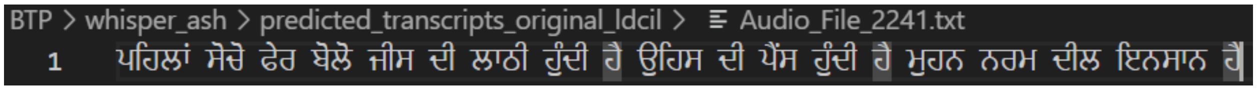
With fine-tuning: WER = 0.55

The snapshots of some of the predicted transcripts are as shown below:

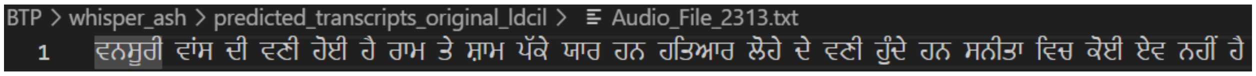
- Predicted Transcripts by original model(without training) on LDCIL dataset:



```
BTP > whisper_ash > predicted_transcripts_original_ldcil > ≡ Audio_File_677.txt
ਮੈਨੂੰ ਯਕੀਨ ਹੈ ਕੀ ਇਕ ਦਿਨ ਸਾਡੇ ਹਾਲਾਦ ਚੂਰ ਟੀਕ ਹੋਣਗੇ ਦੁੱਦ ਜਿੰਚ ਮੱਖੀ ਡੇਗਿਆਈ ਹਰ ਨਾਲ ਕਈਆਈਆਂ ਨੂੰ ਨੁੱਖਸਾਨ ਹੋਇਆਂ ਮੋਰ ਦੇ ਖਾਮਬ ਰੰਘਬਰੰਗੇ ਹੁੰਦੇ ਹਨ
```

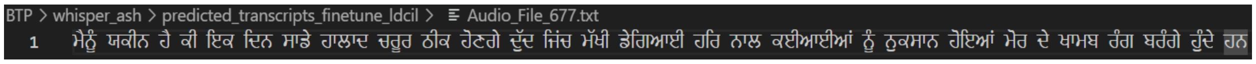


```
BTP > whisper_ash > predicted_transcripts_original_ldcil > ≡ Audio_File_2241.txt
ਪਹਿਲਾਂ ਸੋਚੋ ਫੇਰ ਬੋਲੋ ਜੀਸ ਦੀ ਲਾਠੀ ਹੁੰਦੀ ਹੈ ਉਹਿਸ ਦੀ ਪੈਂਸ ਹੁੰਦੀ ਹੈ ਮੁਹਨ ਨਰਮ ਦੀਲ ਇਨਸਾਨ ਹੈ
```

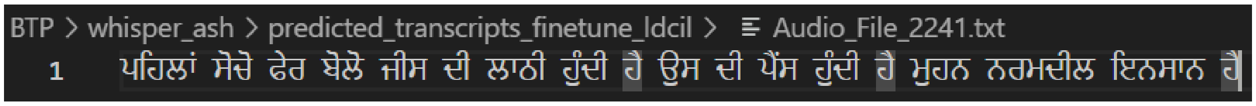


```
BTP > whisper_ash > predicted_transcripts_original_ldcil > ≡ Audio_File_2313.txt
ਵਨਸੁਰੀ ਵਾਂਸ ਦੀ ਵਣੀ ਹੋਈ ਹੈ ਰਾਮ ਤੇ ਸ਼ਾਮ ਪੱਕੇ ਯਾਰ ਹਨ ਹਤਿਆਰ ਲੋਹੇ ਦੇ ਵਣੀ ਹੁੰਦੇ ਹਨ ਸਨੀਤਾ ਵਿਚ ਕੋਈ ਏਵ ਨਹੀਂ ਹੈ
```

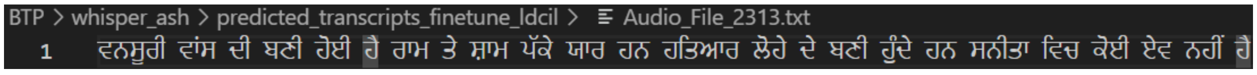
- Predicted Transcripts by fine-tuned model (1 epoch) on LDCIL dataset:



```
BTP > whisper_ash > predicted_transcripts_finetune_ldcil > ≡ Audio_File_677.txt
ਮੈਨੂੰ ਯਕੀਨ ਹੈ ਕੀ ਇਕ ਦਿਨ ਸਾਡੇ ਹਾਲਾਦ ਚਰੂਰ ਠੀਕ ਹੋਣਗੇ ਦੁੱਦ ਜਿੰਚ ਮੱਖੀ ਡੇਗਿਆਈ ਹਰਿ ਨਾਲ ਕਈਆਈਆਂ ਨੂੰ ਨੁਕਸਾਨ ਹੋਇਆਂ ਮੋਰ ਦੇ ਖਾਮਬ ਰੰਗ ਬਰੰਗੇ ਹੁੰਦੇ ਹਨ
```



```
BTP > whisper_ash > predicted_transcripts_finetune_ldcil > ≡ Audio_File_2241.txt
ਪਹਿਲਾਂ ਸੋਚੋ ਫੇਰ ਬੋਲੋ ਜੀਸ ਦੀ ਲਾਠੀ ਹੁੰਦੀ ਹੈ ਉਸ ਦੀ ਪੈਂਸ ਹੁੰਦੀ ਹੈ ਮੁਹਨ ਨਰਮਦੀਲ ਇਨਸਾਨ ਹੈ
```



```
BTP > whisper_ash > predicted_transcripts_finetune_ldcil > ≡ Audio_File_2313.txt
ਵਨਸੁਰੀ ਵਾਂਸ ਦੀ ਬਣੀ ਹੋਈ ਹੈ ਰਾਮ ਤੇ ਸ਼ਾਮ ਪੱਕੇ ਯਾਰ ਹਨ ਹਤਿਆਰ ਲੋਹੇ ਦੇ ਬਣੀ ਹੁੰਦੇ ਹਨ ਸਨੀਤਾ ਵਿਚ ਕੋਈ ਏਵ ਨਹੀਂ ਹੈ
```

Since these results were much better than the results obtained from Wav2Vec, hence we further tested this model on the police data.

The results obtained on police data are as follows:

- <u>Medium_Patiala_Punjabi_med</u>:
- Before fine-tuning: ਹਰੋਹ ਹਾਂ ਹਾਂ ਸ਼ਰੇਆ ਹਾਂ ਹਾਂ ਹਾਂ ਸ਼ਰੇਆ ਹਾਂ ਹਾ
- After fine-tuning: ਅਹਰੋਂ ਅਹਰੋਂ ਆ ਹਾ ਸਰੇਆ ਅਹਨਜੇ ਭਾਰਨਮਸਤੇ ਨੰਮਸਤੇ ਹੋ ਕੈ ਗਰ ਰੀਆਂ ਕੁਛ ਨੀਂ ਭਾਰਬੈਟੇ ਹੋਏ ਥੇ ਅਚਿ ਅਚਿ ਆਮਾਂ ਕਿ ਮਨੂੰ ਪੈਸੇ ਦਭਾਦੰਦੇ ਅਭੀ ਤੋਰ ਦੀਂ ਤੈਨੇ ਤੋਂ ਦਲਵਾਏਂ ਥੇ ਫਰ ਖਤਮ ਓਗਿਂ ਤੇ ਨਾਂ ਬੁਤਾਂ ਅਭੁਮ ਕਾਂ ਤੇ ਦਲਵਾਏਂ ਸਾਈਜੀ ਕੋਵ ਭੀ ਦੁਬਾਰਾਂ ਸੇ ਬੋਲਲਾ ਪੁਢੇਗ ......it is a very long file

- <u>High_Kapurthala_Punjabi_high 1</u>:
- Before fine-tuning: ਅਹਲੋ ਆਂ ਜੀ ਟਰੀ ਗਿੱਦਾਂ ਆ ਠੇਕ ਆ ਬੁੱਦਿ ਵੱਦੀਆ ਵੱਦੀਆ ਬੁਗੇ ਕਿੱਦਾਂ ਸੁਰੇ ਸ਼ਲੇ ਗੀ ਤਾਂ ਫੈਂ ਬੁਗੇ ਵੱਦੀਆ ਇਹਿਂ ਮੈਂ ਗਾ ਜਲਾਂ ਖਲਗਾਰਦਾਂ ਆ ਤੇ ਤ�
- After fine-tuning: ਅਹਲੋ ਆਂ ਜੀ ਡੈਡੀ ਕਿੱਦਾਂ ਆ ਠੇਕ ਆ ਬੁੱਦਰ ਵਧੀਆ ਵੱਧੀਆ ਬੁੰਗੇ ਕਿਹਦਾਂ ਸੁਰੇ ਸ਼ਲੇਗੀ ਤਾਂ ਫਾਂ ਬੁੰਗੇ ਵੱਧੀਆ ਮੈਂ ਗਾ ਜਲਕ ਹਲਗਾਰਦਾਂ ਕਾਹਦੇ ਕਰੀ ਦਾਂ ਫੁਨਾਈ ਟੈਮੱਕ ਕਰੇ ਇਹੁੰਦੇ ਏਨਾ ਮੁੰਮੀ ਤੇਰੀ ਵੀ ਹੋ ਠੀਕ ਆ ਮੁਮਿ ਠੀਕ ਆ ਕਲਵਾਏ ਕਲ ਫੁਨਦਾ ਨਾ ਆਈ ਆਈ ਦੋ ਦੋਵਾਂ ਨਾ ਮਮੋਂ ਦੇ ਦੇ ਕਰ ਲਗਾਣਾ ਅਲੋਗ ਆ ਮਾਦਾ ਗੱਦਾਂ ਤੀਕਿਆ ਤੋਂ ਆਪਨੇ ਛੁਣਾਈ ਬਰਸ ਵੱਦੀਆ ਔਰ ਉਹ ਠੀਕ ਏਂ ਨੇ ਆ ਫਾਂਡੇਵਾਂ ਦੇ ਲੱਗੀ ਸੇ ਉਹ ਤੋਂ ਠੀਕ ਏਂ ਆਂ ਠੀਕ ਏਂ ਮਾਂਦਾ ਔਰ ਹੈਪੀ ਵਣੀ ਠੀਕ ...... it is a very long file

- <u>High_Patiala_Punjabi_high</u>:
- Before fine-tuning: ਹਾਲੋ ਅਹਲੋ ਭਾਭਾ ਗੁਦ ਮੌਨੀਂ ਹਾਲੋ ਹਾਭਾ ਗੁਦ ਮੌਨੀਂ ਗੁਦ ਮੌਨੀਂ ਵੰਚ ਕੀ ਹਲ ਏ ਨਾ ਮਹਿਕ ਕੀ ਆਮੇਂ ਅਚਾ ਮਹਿਕ ਕੀ ਹਲ ਵੱਤ ਠੀਕ ਹੈ ਅੰਜੀ ਨੈ ਠੀਕ ਆ ਵਧੀਂ ਵੱਧੀਂ ਹਿ ਸਕੂਲ ਏ ਅਸ਼ਰ ਤੁੱਭੀ ਗੀ ਫਾਸਰ ਏ ਉਹ ਨੀਂ ਵਤਾਂ ਉਂਹ ਹਾਸਰ ਏ ਚਾਚੁਣਾਂ ਸੋਰ ਅਚਾ ਵਾਂ ਅਥੇ ਕੀ ਅਮਦੇ ਕੂਸ਼ਿਆਲਾਂ ਉਂ�
- After fine-tuning: ਅਹਲੋ ਅਹਲੋ ਭਾਭਾ ਗੁਦ ਮੌਨੀਂ ਅਹਲੋ ਭਾਭਾ ਗੁਦ ਮੌਨੀਂ ਗੁਦ ਮੌਨੀਂ ਵੰਚ ਕੀ ਹਲ ਏ ਨਾ ਮਹਿਕ ਕੀ ਆਮੇਂ ਅਚਾ ਮਹੀਕ ਉਕੇ ਕੇ ਹਾਲਾ ਬੱਤਿ ਠੀਕ ਹੈ? ਆਂ ਜੀ ਨੈਂ ਠੀਕ ਹੈ ਵੱਠੀਂ ਵੱਠੀਂ ਹੈ ਸਕੂਲ ਹੈ? ਆਸ਼ਰ ਤੁੱਦੀ ਗੀ ਫਾਸਰ ਏਹ ਉਹ ਨੀਂ ਵੱਤਾਂ ਉਹ ਹਤਰੇ ਚਾਚੁਣਾਂ ਸੋਰ ਅਚਾ ਵਾਂ ਅਪਥੇ ਕੀਂਦੇ ਕੂਸ਼ਿਆਲਾਂ ਉਹਨ ਸ਼ਾਵਨੂੰ ਜਮਾਂਗ ਸੀ ਖੇਰੇ ਜਾਣਾ ਸੀ ਤੂੰਦ ਔਰ ਛਿ ਮੁਮੀਰਾਂ ਦੇ ਰਾਂਦੋ ਨੇ ਸਕੁਲ ਕਿਨੇ ਵਿਦਾ ਟੈਮ ਉਗਿਆ ਉਣ ਉਨ੍ਹਿਆ ਉਝੇ ਬਾਰਾ ਵੱਜੁ ਦੇ ਹੀ ਆਈ ਬਾਰਾ ਤੋਂ ਪਾਂਝ ਬਾਰਾ ਤੋਂ ਤੀਨ ਤੇ ਪੱਚ ਚੰਲ ਰੇ ਲੇ ਏਂਚਿ ਤਾਂ ਏਂ ਮਿਚਾਰ ਰੇਂ ਨੇ ਫਿਰ ਠੀਕ ਹੈ ਦੋਣਾਂ ...... it is a very long file

- <u>Low_Patiala_Punjabi_low</u>:
- Before fine-tuning: ਵੈਲੋ ਹਾਂਜੇ
- After fine-tuning: ਵੈਲੋ ਹਾਂਜੇ ਵਦੀਆ ਵਦੀਆ ਕੇਮੇਆਂ ਭਾਂ ਦਬਾਓ ਦਲਾਂ ਕਦੀਰ ਕਦੀਆਂ ਦੇ ਬਡੈ ਵਰਾਲਾ ਖੋਰ ਬਾਂ ਚੀਤ ਨਮਨ ਤੁੱਛ ਕਿ ਦੱਤ ਤੋਂ ਭਾਂ ਚੇਤ ਨਾਂ ਭਾਂ ਚੇਤ ਨਾਂ ਹੋਹ ਸਨਾਂ ਕਨੀਆਂ ਜਾਏਂ ਕਿ ਵੈਲਵਾਰ ਹੋਰ ਮੇਤੁ ਮੀਖੂਆਵੇ ਇੱਤੀ ਇਹਦੀ ਆਂ ਜੀ ਆਂ ਜਾਂਦੇ ਆ ਆਹੋਂ ਮੈਂ ਪੈਂਡ ਪਹੁੰਚ ਗੀ ਸੀ ਤੋਂ ਮੈਂ ਸਿਰੇ ਮੈਂ ਪੈਂਡ ਪਹੁੰ ਜਿਆਂਦ ਭਾਬੀ ਤਿਰੀ ਨਾਲਾ ਤਣਿ ਨਗਾਂ ਲਿੰਦਾ ਮੈਂ ਸਿਰੇ ਮੈਂ ਤਾਂ ਪੈਂਡ ਭਾਉਂ ਤਿਰੀ ਹਨ ਦੀ ਤੁਸੀ ਬੰਦਰਾਂ ਤੋਂ ਬਾਅਦ ਹੀ ਰੱਖੋ ਨ ਪਰਾਵੀ ਕੋਰੀ ਸੌਂਡ ਦੇ ਨਹੀਂ ਅਤੇ ਸੱਲੇ ਸੁਲੇ ਕੋਈ ਮੰਗਾਲੇ ਨਹੀਂ ਸ਼ੁਰੇ ਨੂੰ ਫੋਨਪਾਨ ਲਾਰਹੇ ਉਹਨੂੰ ਪੁੱਛਲੇ ਨੂੰ ਜੇ ਚੱਕਲੇ ਬੈਂਟੋ ਕੇ ਉਹ ਰੈਟਿ ਜੀ ਲਾਈ ਸੀ ਤੀਂਗਣਿਆਉਦਾ ਅੰਜੀ ਮਾ ਵੇਖਲ

ਉਂਗਲਾਕ ਲਾਇਆ ਸੀਂਲਾਂ ਨੂੰ ਪਫਾਂ ਕਿ ਸਾਉਣ ਲਾਇਆ ਸੀਂਲਾਂ ਬੀਜੀ ਜਾਂਡ ਨੰਫੋਤ ਏ ਚੱਲ ਕਰਦਾਂ ਫਿਰ ਮੈ ਸੋਬਾ ਪਰੋਗਿ

- <u>Low_Ropar_Punjabi_low</u>:
- Before fine-tuning: ਆਲੋਂ ਆਲੋਂ ਆਲੋਂ ਆਲੋਂ ਆਲੋਂ ਆਲੋਂ ਆਲੋਂ ਆਲੋਂ ਆਲੋਂ ਆਲੋਂ ਆਲੋਂ ਆਲੋਂ ਆਲੋਂ ਆਲੋਂ ਆਲੋਂ ਆਲੋਂ ਆਲੋਂ ਆਲੋਂ
- After fine-tuning: ਅਲੋਂ ਆਂ ਅਵਾਂ ਠੀਕ ਠਾਗ ਆਂ ਦੇ ਜੇਖ ਤੋਂ ਆਪਣਾਂ ਸੁਣਾਂ ਭਾਲ ਉਦਾ ਬਿਆਂ ਵੀ ਉਦਾ ਗਿੰਦੇ ਦੇ ਭਾਈ ਦਾ ਉਤਰ ਗੇ ਸੰਤ ਉਤਰ ਮੁੰਟੇ ਦਾ ਛੋਤੇ ਦਾ ਆਲੋਂ ਆਲੋਂ ਆਲੋਂ ਆਲੋਂ ਆਲੋਂ ਆਲੋਂ ਆਲੋਂ ਆਲੋਂ ਆਲੋਂ ਆਲੋਂ ਆਲੋਂ ਆਲੋਂ ਆਲੋਂ ਆਲੋਂ ਆਲੋਂ ਆਲੋਂ ਆਲ ਵਾਂ ਵਾਂ ਜੁੱਕ ਨਾ ਪਾਣੀ ਖਿੱਟਿਆ ਵਾਂ ਵਹਿਲੇ ਏਂ ਤੁਮਾਲ ਵਹਿਲੇ ਉੱਤੇ ਏਂ ਵਜੂਡ ਵਜੂਡ ਤੋਂ ਪਾਟ ਆਇਆਂ ਤੇ ਏਂ ਫੇਲ ਅਚਿਆ ਹੋ ਠੀਕ ਹੈ ਆਂ ਵੱਦੀਆ ਭਹਦਵਾਂ ਮਾਂ ਥੀਕ ਠਾਇੰ ਦੈ ਵਦੀਆ ਤੂੰ ਸਨਾਂ ਵਾਲਿ ਉਤਾਲਜੀ ਦਾ ਪੁਛਲੇ ਰਾਲ ਚਲ ਮਾਂ ਮਾਂ ਚੱਲ੍ਹ ਕਿਲ੍ਹ ਕਰਦਾ ਫੋਂ ਭਾਂ ਵੱਤਾ ਧੀਗ ਹੋਲ ਹਲ� ਵੱਲੋ ਤਾਂ ਗਿਆਂਸਰ ਦੀ ਵੀਚ ਹੋਣ ਨੋਂ ਟਾਂ ਵਾ ਠੀਟਾਂ ਦਾ ਚੰਦਾਂ ਹਿਲਾਂ ਚੰਦਾਂ

4. GUI and Keyword Search

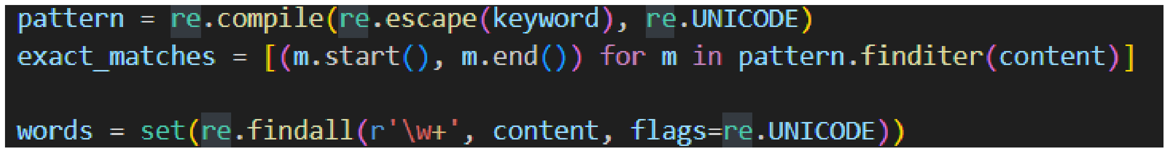
We have made a basic GUI using tkinter which takes a Text File as input, and provides options for keyword search (exact or approx) and semantic search (same meaning words/sentences implying same meaning) for checking if the file is suspicious. Exact matches are highlighted green, approx matches are highlighted yellow.

4.1 KEYWORD SEARCH

We have used three different types of keyword search:

4.1.1 Re (exact keyword match):

The ***re module*** in Python provides support for regular expressions (regex), which are a language for matching text patterns.



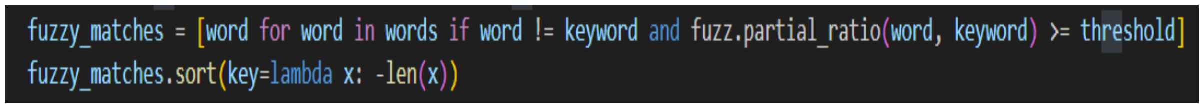
```
pattern = re.compile(re.escape(keyword), re.UNICODE)
exact_matches = [(m.start(), m.end()) for m in pattern.finditer(content)]

words = set(re.findall(r'\w+', content, flags=re.UNICODE))
```

It matches each character from keyword to every word (all positions) in the sentences of text file.

4.1.2 Fuzzywuzzy (partial keyword match):

FuzzyWuzzy is a Python library used for fuzzy string matching, which helps find approximate matches between strings using a threshold (here 0.8)

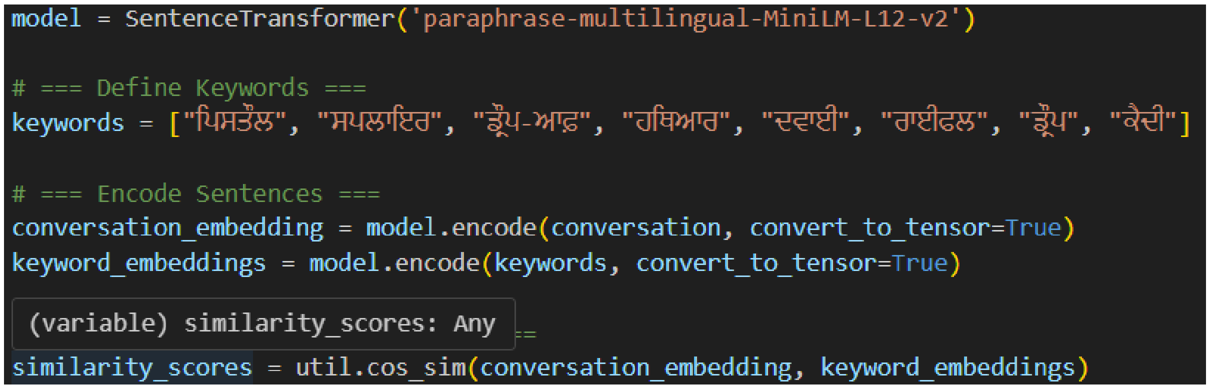


```
fuzzy_matches = [word for word in words if word != keyword and fuzz.partial_ratio(word, keyword) >= threshold]
fuzzy_matches.sort(key=lambda x: -len(x))
```

It uses Levenshtein distance which is number of single-character edits: insertions, deletions or substitutions, are needed to change ane string into another.

4.1.3 Semantic Keyword Match:

It is used for same meaning keyword match (gives scores for each keyword depending on its relevance with the text file, even if the same words are not present).


```
model = SentenceTransformer('paraphrase-multilingual-MiniLM-L12-v2')

# === Define Keywords ===
keywords = ["ਪਿਸਤੌਲ", "ਸਪਲਾਇਰ", "ਡ੍ਰੌਪ-ਆਫ਼", "ਹਥਿਆਰ", "ਦਵਾਈ", "ਰਾਈਫਲ", "ਡ੍ਰੌਪ", "ਕੈਦੀ"]

# === Encode Sentences ===
conversation_embedding = model.encode(conversation, convert_to_tensor=True)
keyword_embeddings = model.encode(keywords, convert_to_tensor=True)

 (variable) similarity_scores: Any ==
similarity_scores = util.cos_sim(conversation_embedding, keyword_embeddings)
```

It uses paraphrase-multilingual-MiniLM-L12-v2 (hugging face, present in sentence_transformers) which calculates Cosine Similarity between vectors (embeddings). This model is trained on over **50+ languages**, including English, Hindi, Punjabi, Urdu, Arabic, German, French, etc. using **Multilingual Parallel Corpora** (e.g., from Wikipedia, Common Crawl, Tatoeba, and OpenSubtitles)

4.2 GUI

Step-by-step instructions to run the GUI:

Run the python code on your local machine (we used VScode). A new window (GUI) opens in 8-10 seconds (as models need to be called).

- **First page** has a "Select File" option. Select the Punjabi text file from the local machine.
- As soon as you select the file, the **second page** for Exact and Fuzzy (approx) keyword search opens. It has:
  - "Search" bar and button, where you can enter any number of words separated by comma.
  - Boxes displaying number of exact and fuzzy matches for given keywords (new box for each keyword).
  - Two text files Original and Highlighted (where exact and fuzzy matches are highlighted green and yellow respectively) with scroll bars.
  - "Save Output" button to save the highlighted text file (in the same folder as the original file, with name originalFileName_text.txt). Here exact matches are underlined, and fuzzy matches are in bold.
  - "Back" button to go to the first page and select a new file.
  - "Semantic Search" button to go to the next page for semantic search on the same selected text file.

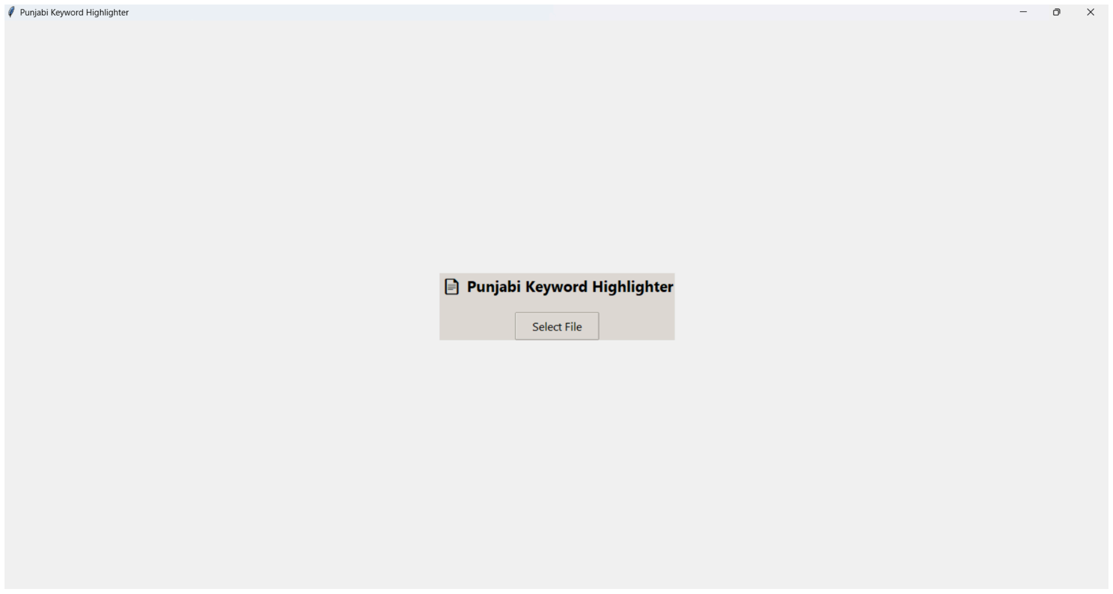
- As soon as you select "Semantic Search", the **third page** for Semantic Search opens. It has:
  - "Search" bar where you can enter any number of words separated by comma.
  - "Run Semantic Search" button to run semantic analysis of entered keywords on the text file selected in the first page. It might take time depending on the number of keywords entered and the length of the text file.
  - Text Box with scroll bar to print how much the text file implies meaning related to the entered keywords. It gives scores for each keyword (>0.3 is a good match) along with a number of exact and fuzzy matches in brackets.
  - "Back" button to go to the second page.

Note:

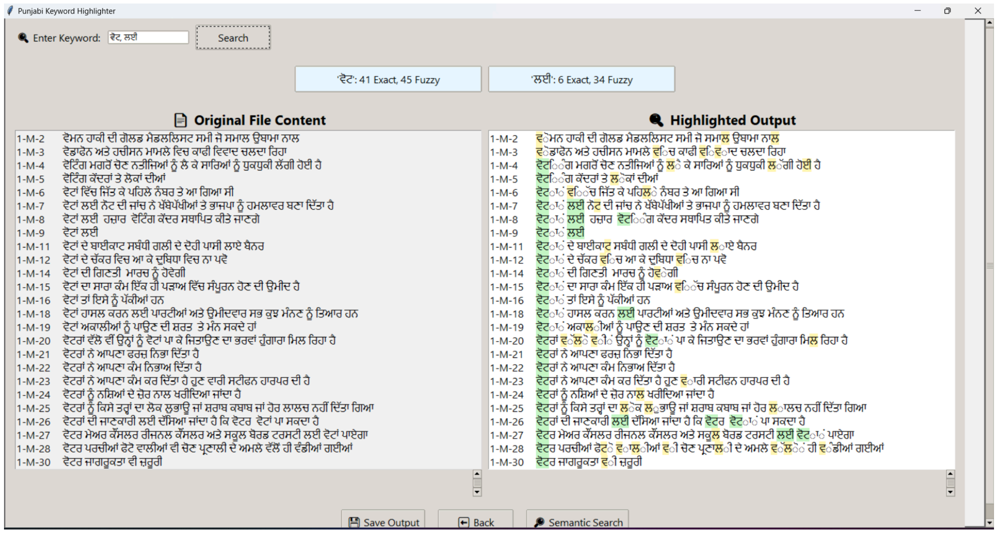
➔ Default keyword for Second page is "ਵੋਟ". This can be changed.

➔ Default keywords for Third page are:
"ਪਿਸਤੌਲ, ਸਪਲਾਇਰ, ਡ੍ਰੌਪ-ਆਫ਼, ਹਥਿਆਰ, ਦਵਾਈ, ਰਾਈਫਲ, ਡ੍ਰੌਪ, ਕੈਦੀ"
These can also be changed.

Step-by-step screenshots of the GUI for reference:



File selected: M1-M50.txt


Punjabi Keyword Highlighter

Enter Keyword: ਵੋਟ, ਲਈ

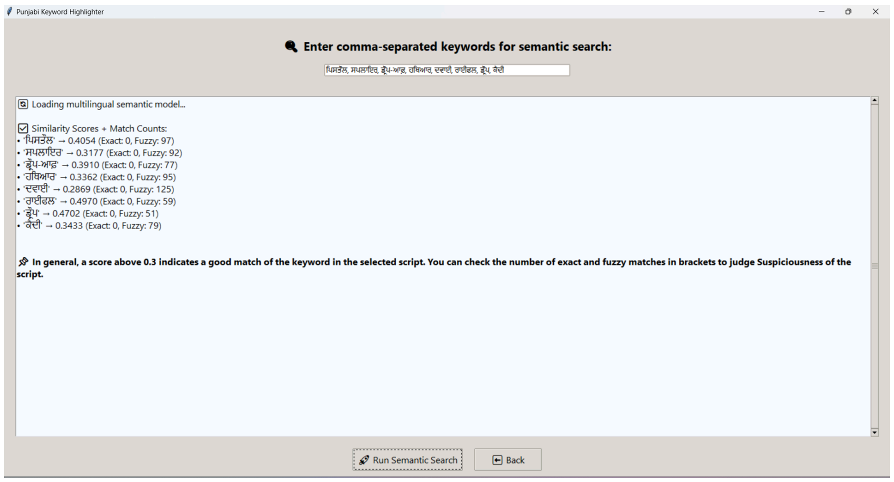
'ਵੋਟ': 41 Exact, 45 Fuzzy

'ਲਈ': 6 Exact, 34 Fuzzy

Original File Content

```
1-M-2    ਵੋਮਨ ਹਾਕੀ ਦੀ ਗੋਲਡ ਮੈਡਲਲਿਸਟ ਸਮੀ ਜੋ ਸਮਾਲ ਉਬਾਮਾ ਨਾਲ
1-M-3    ਵੋਡਾਫੋਨ ਅਤੇ ਹਚੀਸਨ ਮਾਮਲੇ ਵਿਚ ਕਾਫੀ ਵਿਵਾਦ ਚਲਦਾ ਰਿਹਾ
1-M-4    ਵੋਟਿੰਗ ਮਗਰੋਂ ਚੋਣ ਨਤੀਜਿਆਂ ਨੂੰ ਲੈ ਕੇ ਸਾਰਿਆਂ ਨੂੰ ਧੁਕਧੁਕੀ ਲੱਗੀ ਹੋਈ ਹੈ
1-M-5    ਵੋਟਿੰਗ ਕੇਂਦਰਾਂ ਤੇ ਲੋਕਾਂ ਦੀਆਂ
1-M-6    ਵੋਟਾਂ ਵਿੱਚ ਜਿੱਤ ਕੇ ਪਹਿਲੇ ਨੰਬਰ ਤੇ ਆ ਗਿਆ ਸੀ
1-M-7    ਵੋਟਾਂ ਲਈ ਨੋਟ ਦੀ ਜਾਂਚ ਨੇ ਖੱਬੇਪੱਖੀਆਂ ਤੇ ਭਾਜਪਾ ਨੂੰ ਹਮਲਾਵਰ ਬਣਾ ਦਿੱਤਾ ਹੈ
1-M-8    ਵੋਟਾਂ ਲਈ  ਹਜ਼ਾਰ  ਵੋਟਿੰਗ ਕੇਂਦਰ ਸਥਾਪਿਤ ਕੀਤੇ ਜਾਣਗੇ
1-M-9    ਵੋਟਾਂ ਲਈ
1-M-11   ਵੋਟਾਂ ਦੇ ਬਾਈਕਾਟ ਸਬੰਧੀ ਗਲੀ ਦੇ ਦੋਹੀ ਪਾਸੀ ਲਾਏ ਬੈਨਰ
1-M-12   ਵੋਟਾਂ ਦੇ ਚੱਕਰ ਵਿਚ ਆ ਕੇ ਦੁਬਿਧਾ ਵਿਚ ਨਾ ਪਵੋ
1-M-14   ਵੋਟਾਂ ਦੀ ਗਿਣਤੀ  ਮਾਰਚ ਨੂੰ ਹੋਵੇਗੀ
1-M-15   ਵੋਟਾਂ ਦਾ ਸਾਰਾ ਕੰਮ ਇੱਕ ਹੀ ਪੜਾਅ ਵਿੱਚ ਸੰਪੂਰਨ ਹੋਣ ਦੀ ਉਮੀਦ ਹੈ
1-M-16   ਵੋਟਾਂ ਤਾਂ ਇਸੇ ਨੂੰ ਪੱਕੀਆਂ ਹਨ
1-M-18   ਵੋਟਾਂ ਹਾਸਲ ਕਰਨ ਲਈ ਪਾਰਟੀਆਂ ਅਤੇ ਉਮੀਦਵਾਰ ਸਭ ਕੁਝ ਮੰਨਣ ਨੂੰ ਤਿਆਰ ਹਨ
1-M-19   ਵੋਟਾਂ ਅਕਾਲੀਆਂ ਨੂੰ ਪਾਉਣ ਦੀ ਸ਼ਰਤ  ਤੇ ਮੰਨ ਸਕਦੇ ਹਾਂ
1-M-20   ਵੋਟਰਾਂ ਵੱਲੋ ਵੀਂ ਉਨ੍ਹਾਂ ਨੂੰ ਵੋਟਾਂ ਪਾ ਕੇ ਜਿਤਾਉਣ ਦਾ ਭਰਵਾਂ ਹੁੰਗਾਰਾ ਮਿਲ ਰਿਹਾ ਹੈ
1-M-21   ਵੋਟਰਾਂ ਨੇ ਆਪਣਾ ਫਰਜ਼ ਨਿਭਾ ਦਿੱਤਾ ਹੈ
1-M-22   ਵੋਟਰਾਂ ਨੇ ਆਪਣਾ ਕੰਮ ਨਿਭਾਅ ਦਿੱਤਾ ਹੈ
1-M-23   ਵੋਟਰਾਂ ਨੇ ਆਪਣਾ ਕੰਮ ਕਰ ਦਿੱਤਾ ਹੈ ਹੁਣ ਵਾਰੀ ਸਟੀਫਨ ਹਾਰਪਰ ਦੀ ਹੈ
1-M-24   ਵੋਟਰਾਂ ਨੂੰ ਨਸ਼ਿਆਂ ਦੇ ਜ਼ੋਰ ਨਾਲ ਖਰੀਦਿਆ ਜਾਂਦਾ ਹੈ
1-M-25   ਵੋਟਰਾਂ ਨੂੰ ਕਿਸੇ ਤਰ੍ਹਾਂ ਦਾ ਲੋਕ ਲੁਭਾਊ ਜਾਂ ਸ਼ਰਾਬ ਕਬਾਬ ਜਾਂ ਹੋਰ ਲਾਲਚ ਨਹੀਂ ਦਿੱਤਾ ਗਿਆ
1-M-26   ਵੋਟਰਾਂ ਦੀ ਜਾਣਕਾਰੀ ਲਈ ਦੱਸਿਆ ਜਾਂਦਾ ਹੈ ਕਿ ਵੋਟਰ  ਵੋਟਾਂ ਪਾ ਸਕਦਾ ਹੈ
1-M-27   ਵੋਟਰ ਮੇਅਰ ਕੌਂਸਲਰ ਰੀਜਨਲ ਕੌਂਸਲਰ ਅਤੇ ਸਕੂਲ ਬੋਰਡ ਟਰਸਟੀ ਲਈ ਵੋਟਾਂ ਪਾਏਗਾ
1-M-28   ਵੋਟਰ ਪਰਚੀਆਂ ਫੋਟੋ ਵਾਲੀਆਂ ਵੀ ਚੋਣ ਪ੍ਰਣਾਲੀ ਦੇ ਅਮਲੇ ਵੱਲੋਂ ਹੀ ਵੰਡੀਆਂ ਗਈਆਂ
1-M-30   ਵੋਟਰ ਜਾਗਰੂਕਤਾ ਵੀ ਜ਼ਰੂਰੀ
```

Highlighted Output


Punjabi Keyword Highlighter

Enter comma-separated keywords for semantic search:

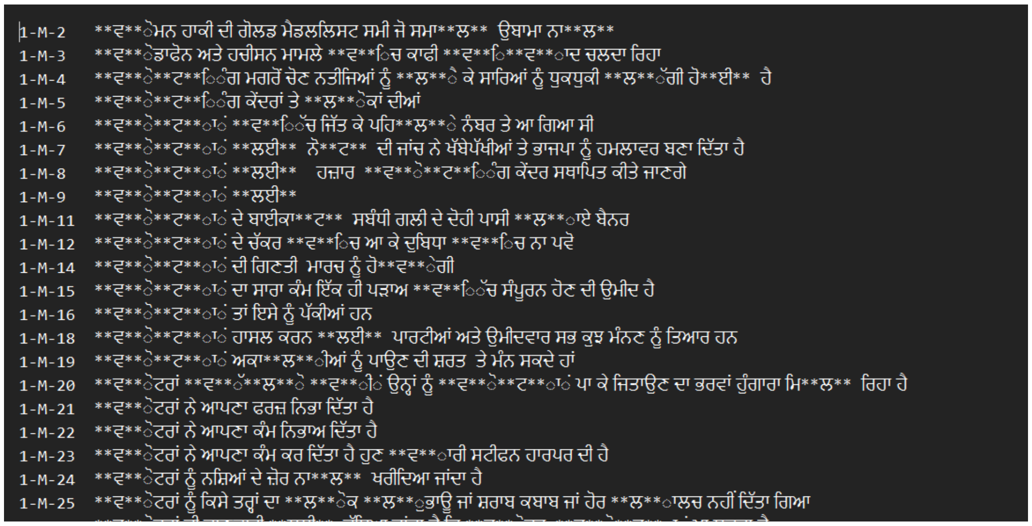
ਪਿਸਤੌਲ, ਸਪਲਾਇਰ, ਡ੍ਰੌਪ-ਆਫ਼, ਹਥਿਆਰ, ਦਵਾਈ, ਰਾਈਫਲ, ਡ੍ਰੌਪ, ਕੈਦੀ

```
Loading multilingual semantic model...

Similarity Scores + Match Counts:
• 'ਪਿਸਤੌਲ' → 0.4054 (Exact: 0, Fuzzy: 97)
• 'ਸਪਲਾਇਰ' → 0.3177 (Exact: 0, Fuzzy: 92)
• 'ਡ੍ਰੌਪ-ਆਫ਼' → 0.3910 (Exact: 0, Fuzzy: 77)
• 'ਹਥਿਆਰ' → 0.3362 (Exact: 0, Fuzzy: 95)
• 'ਦਵਾਈ' → 0.2869 (Exact: 0, Fuzzy: 125)
• 'ਰਾਈਫਲ' → 0.4970 (Exact: 0, Fuzzy: 59)
• 'ਡ੍ਰੌਪ' → 0.4702 (Exact: 0, Fuzzy: 51)
• 'ਕੈਦੀ' → 0.3433 (Exact: 0, Fuzzy: 79)
```

In general, a score above 0.3 indicates a good match of the keyword in the selected script. You can check the number of exact and fuzzy matches in brackets to judge Suspiciousness of the script.

Text File saved (M1-M50_text.txt):


```
1-M-2    **ਵ**ੋਮਨ ਹਾਕੀ ਦੀ ਗੋਲਡ ਮੈਡਲਲਿਸਟ ਸਮੀ ਜੋ ਸਮਾ**ਲ**  ਉਬਾਮਾ ਨਾ**ਲ**
1-M-3    **ਵ**ੋਡਾਫੋਨ ਅਤੇ ਹਚੀਸਨ ਮਾਮਲੇ **ਵ**ਿਚ ਕਾਫੀ **ਵ**ਿ**ਵ**ਾਦ ਚਲਦਾ ਰਿਹਾ
1-M-4    **ਵ**ੋ**ਟ**ਿੰਗ ਮਗਰੋਂ ਚੋਣ ਨਤੀਜਿਆਂ ਨੂੰ **ਲ**ੈ ਕੇ ਸਾਰਿਆਂ ਨੂੰ ਧੁਕਧੁਕੀ **ਲ**ੱਗੀ ਹੋ**ਈ**  ਹੈ
1-M-5    **ਵ**ੋ**ਟ**ਿੰਗ ਕੇਂਦਰਾਂ ਤੇ **ਲ**ੋਕਾਂ ਦੀਆਂ
1-M-6    **ਵ**ੋ**ਟ**ਾਂ **ਵ**ਿੱਚ ਜਿੱਤ ਕੇ ਪਹਿ**ਲ**ੇ ਨੰਬਰ ਤੇ ਆ ਗਿਆ ਸੀ
1-M-7    **ਵ**ੋ**ਟ**ਾਂ **ਲਈ**  ਨੋ**ਟ**  ਦੀ ਜਾਂਚ ਨੇ ਖੱਬੇਪੱਖੀਆਂ ਤੇ ਭਾਜਪਾ ਨੂੰ ਹਮਲਾਵਰ ਬਣਾ ਦਿੱਤਾ ਹੈ
1-M-8    **ਵ**ੋ**ਟ**ਾਂ **ਲਈ**    ਹਜ਼ਾਰ  **ਵ**ੋ**ਟ**ਿੰਗ ਕੇਂਦਰ ਸਥਾਪਿਤ ਕੀਤੇ ਜਾਣਗੇ
1-M-9    **ਵ**ੋ**ਟ**ਾਂ **ਲਈ**
1-M-11   **ਵ**ੋ**ਟ**ਾਂ ਦੇ ਬਾਈਕਾ**ਟ**  ਸਬੰਧੀ ਗਲੀ ਦੇ ਦੋਹੀ ਪਾਸੀ **ਲ**ਾਏ ਬੈਨਰ
1-M-12   **ਵ**ੋ**ਟ**ਾਂ ਦੇ ਚੱਕਰ **ਵ**ਿਚ ਆ ਕੇ ਦੁਬਿਧਾ **ਵ**ਿਚ ਨਾ ਪਵੋ
1-M-14   **ਵ**ੋ**ਟ**ਾਂ ਦੀ ਗਿਣਤੀ  ਮਾਰਚ ਨੂੰ ਹੋ**ਵ**ੇਗੀ
1-M-15   **ਵ**ੋ**ਟ**ਾਂ ਦਾ ਸਾਰਾ ਕੰਮ ਇੱਕ ਹੀ ਪੜਾਅ **ਵ**ਿੱਚ ਸੰਪੂਰਨ ਹੋਣ ਦੀ ਉਮੀਦ ਹੈ
1-M-16   **ਵ**ੋ**ਟ**ਾਂ ਤਾਂ ਇਸੇ ਨੂੰ ਪੱਕੀਆਂ ਹਨ
1-M-18   **ਵ**ੋ**ਟ**ਾਂ ਹਾਸਲ ਕਰਨ **ਲਈ**  ਪਾਰਟੀਆਂ ਅਤੇ ਉਮੀਦਵਾਰ ਸਭ ਕੁਝ ਮੰਨਣ ਨੂੰ ਤਿਆਰ ਹਨ
1-M-19   **ਵ**ੋ**ਟ**ਾਂ ਅਕਾ**ਲ**ੀਆਂ ਨੂੰ ਪਾਉਣ ਦੀ ਸ਼ਰਤ  ਤੇ ਮੰਨ ਸਕਦੇ ਹਾਂ
1-M-20   **ਵ**ੋਟਰਾਂ **ਵ**ੱ**ਲ**ੋ **ਵ**ੀਂ ਉਨ੍ਹਾਂ ਨੂੰ **ਵ**ੋ**ਟ**ਾਂ ਪਾ ਕੇ ਜਿਤਾਉਣ ਦਾ ਭਰਵਾਂ ਹੁੰਗਾਰਾ ਮਿ**ਲ**  ਰਿਹਾ ਹੈ
1-M-21   **ਵ**ੋਟਰਾਂ ਨੇ ਆਪਣਾ ਫਰਜ਼ ਨਿਭਾ ਦਿੱਤਾ ਹੈ
1-M-22   **ਵ**ੋਟਰਾਂ ਨੇ ਆਪਣਾ ਕੰਮ ਨਿਭਾਅ ਦਿੱਤਾ ਹੈ
1-M-23   **ਵ**ੋਟਰਾਂ ਨੇ ਆਪਣਾ ਕੰਮ ਕਰ ਦਿੱਤਾ ਹੈ ਹੁਣ **ਵ**ਾਰੀ ਸਟੀਫਨ ਹਾਰਪਰ ਦੀ ਹੈ
1-M-24   **ਵ**ੋਟਰਾਂ ਨੂੰ ਨਸ਼ਿਆਂ ਦੇ ਜ਼ੋਰ ਨਾ**ਲ**  ਖਰੀਦਿਆ ਜਾਂਦਾ ਹੈ
1-M-25   **ਵ**ੋਟਰਾਂ ਨੂੰ ਕਿਸੇ ਤਰ੍ਹਾਂ ਦਾ **ਲ**ੋਕ **ਲ**ੁਭਾਊ ਜਾਂ ਸ਼ਰਾਬ ਕਬਾਬ ਜਾਂ ਹੋਰ **ਲ**ਾਲਚ ਨਹੀਂ ਦਿੱਤਾ ਗਿਆ
```

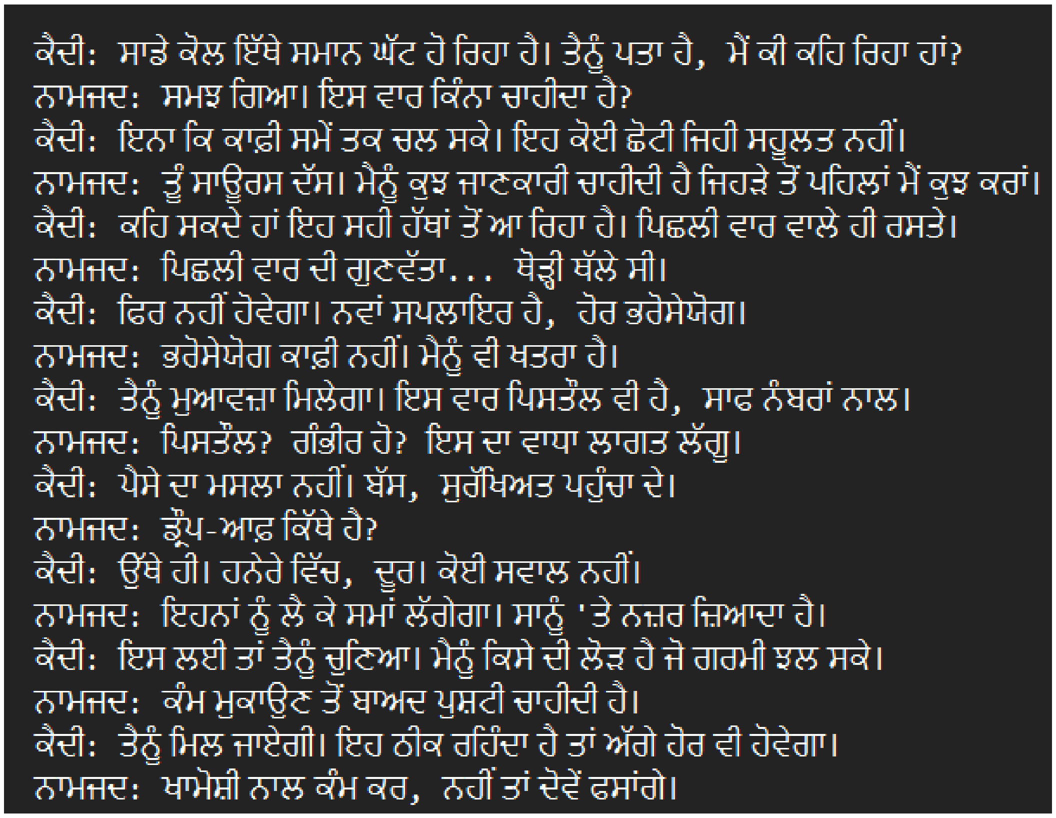
5. Suspicious Check

- Semantic search combined with Exact and Fuzzy can be used for checking if a given conversation in transcripts is suspicious

- For eg: We can check the relevance of **words like Gun, Police, Riffle, Bomb, knife etc** with given text file using Semantic search, and also get exact+fuzzy count

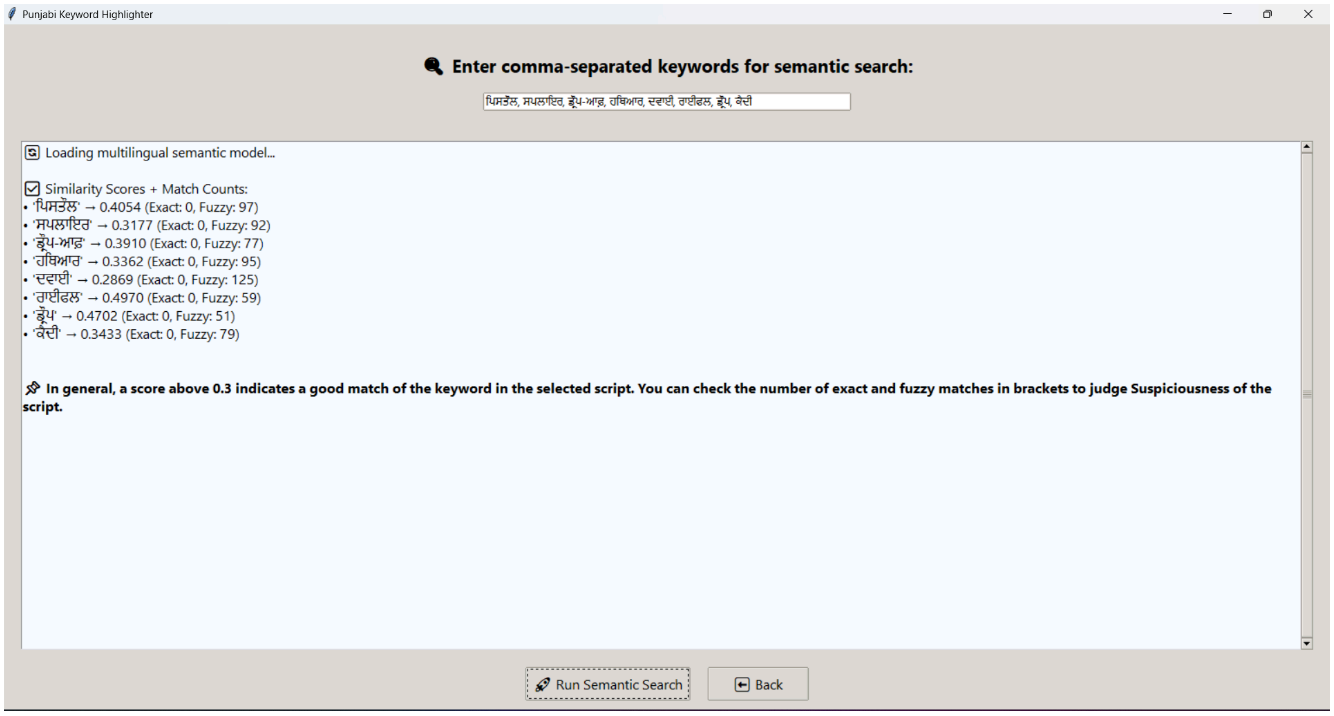
- This feature has been included in the GUI, and the user can design his own keyword library (list) to be checked in the selected text file
  By default, these are set as:
  “ਪਿਸਤੌਲ, ਸਪਲਾਇਰ, ਡ੍ਰੌਪ-ਆਫ਼, ਹਥਿਆਰ, ਦਵਾਈ, ਰਾਈਫਲ, ਡ੍ਰੌਪ, ਕੈਦੀ”

Example:
Text file:



ਕੈਦੀ: ਸਾਡੇ ਕੋਲ ਇੱਥੇ ਸਮਾਨ ਘੱਟ ਹੋ ਰਿਹਾ ਹੈ। ਤੈਨੂੰ ਪਤਾ ਹੈ, ਮੈਂ ਕੀ ਕਹਿ ਰਿਹਾ ਹਾਂ?
ਨਾਮਜਦ: ਸਮਝ ਗਿਆ। ਇਸ ਵਾਰ ਕਿੰਨਾ ਚਾਹੀਦਾ ਹੈ?
ਕੈਦੀ: ਇਨਾ ਕਿ ਕਾਫ਼ੀ ਸਮੇਂ ਤਕ ਚਲ ਸਕੇ। ਇਹ ਕੋਈ ਛੋਟੀ ਜਿਹੀ ਸਹੂਲਤ ਨਹੀਂ।
ਨਾਮਜਦ: ਤੂੰ ਸਾਊਰਸ ਦੱਸ। ਮੈਨੂੰ ਕੁਝ ਜਾਣਕਾਰੀ ਚਾਹੀਦੀ ਹੈ ਜਿਹੜੇ ਤੋਂ ਪਹਿਲਾਂ ਮੈਂ ਕੁਝ ਕਰਾਂ।
ਕੈਦੀ: ਕਹਿ ਸਕਦੇ ਹਾਂ ਇਹ ਸਹੀ ਹੱਥਾਂ ਤੋਂ ਆ ਰਿਹਾ ਹੈ। ਪਿਛਲੀ ਵਾਰ ਵਾਲੇ ਹੀ ਰਸਤੇ।
ਨਾਮਜਦ: ਪਿਛਲੀ ਵਾਰ ਦੀ ਗੁਣਵੱਤਾ... ਥੋੜ੍ਹੀ ਥੱਲੇ ਸੀ।
ਕੈਦੀ: ਫਿਰ ਨਹੀਂ ਹੋਵੇਗਾ। ਨਵਾਂ ਸਪਲਾਇਰ ਹੈ, ਹੋਰ ਭਰੋਸੇਯੋਗ।
ਨਾਮਜਦ: ਭਰੋਸੇਯੋਗ ਕਾਫ਼ੀ ਨਹੀਂ। ਮੈਨੂੰ ਵੀ ਖਤਰਾ ਹੈ।
ਕੈਦੀ: ਤੈਨੂੰ ਮੁਆਵਜ਼ਾ ਮਿਲੇਗਾ। ਇਸ ਵਾਰ ਪਿਸਤੌਲ ਵੀ ਹੈ, ਸਾਫ ਨੰਬਰਾਂ ਨਾਲ।
ਨਾਮਜਦ: ਪਿਸਤੌਲ? ਗੰਭੀਰ ਹੋ? ਇਸ ਦਾ ਵਾਧਾ ਲਾਗਤ ਲੱਗੂ।
ਕੈਦੀ: ਪੈਸੇ ਦਾ ਮਸਲਾ ਨਹੀਂ। ਬੱਸ, ਸੁਰੱਖਿਅਤ ਪਹੁੰਚਾ ਦੇ।
ਨਾਮਜਦ: ਡ੍ਰੌਪ-ਆਫ਼ ਕਿੱਥੇ ਹੈ?
ਕੈਦੀ: ਉੱਥੇ ਹੀ। ਹਨੇਰੇ ਵਿੱਚ, ਦੂਰ। ਕੋਈ ਸਵਾਲ ਨਹੀਂ।
ਨਾਮਜਦ: ਇਹਨਾਂ ਨੂੰ ਲੈ ਕੇ ਸਮਾਂ ਲੱਗੇਗਾ। ਸਾਨੂੰ 'ਤੇ ਨਜ਼ਰ ਜ਼ਿਆਦਾ ਹੈ।
ਕੈਦੀ: ਇਸ ਲਈ ਤਾਂ ਤੈਨੂੰ ਚੁਣਿਆ। ਮੈਨੂੰ ਕਿਸੇ ਦੀ ਲੋੜ ਹੈ ਜੋ ਗਰਮੀ ਝੱਲ ਸਕੇ।
ਨਾਮਜਦ: ਕੰਮ ਮੁਕਾਉਣ ਤੋਂ ਬਾਅਦ ਪੁਸ਼ਟੀ ਚਾਹੀਦੀ ਹੈ।
ਕੈਦੀ: ਤੈਨੂੰ ਮਿਲ ਜਾਏਗੀ। ਇਹ ਠੀਕ ਰਹਿੰਦਾ ਹੈ ਤਾਂ ਅੱਗੇ ਹੋਰ ਵੀ ਹੋਵੇਗਾ।
ਨਾਮਜਦ: ਖਾਮੋਸ਼ੀ ਨਾਲ ਕੰਮ ਕਰ, ਨਹੀਂ ਤਾਂ ਦੋਵੇਂ ਫਸਾਂਗੇ।

After Semantic Search on it:



Enter comma-separated keywords for semantic search:

ਪਿਸਤੌਲ, ਸਪਲਾਇਰ, ਡ੍ਰੌਪ-ਆਫ਼, ਹਥਿਆਰ, ਦਵਾਈ, ਰਾਈਫਲ, ਡ੍ਰੌਪ, ਕੈਦੀ

```
Loading multilingual semantic model...

Similarity Scores + Match Counts:
• 'ਪਿਸਤੌਲ' → 0.4054 (Exact: 0, Fuzzy: 97)
• 'ਸਪਲਾਇਰ' → 0.3177 (Exact: 0, Fuzzy: 92)
• 'ਡ੍ਰੌਪ-ਆਫ਼' → 0.3910 (Exact: 0, Fuzzy: 77)
• 'ਹਥਿਆਰ' → 0.3362 (Exact: 0, Fuzzy: 95)
• 'ਦਵਾਈ' → 0.2869 (Exact: 0, Fuzzy: 125)
• 'ਰਾਈਫਲ' → 0.4970 (Exact: 0, Fuzzy: 59)
• 'ਡ੍ਰੌਪ' → 0.4702 (Exact: 0, Fuzzy: 51)
• 'ਕੈਦੀ' → 0.3433 (Exact: 0, Fuzzy: 79)
```

In general, a score above 0.3 indicates a good match of the keyword in the selected script. You can check the number of exact and fuzzy matches in brackets to judge Suspiciousness of the script.

Thus we can conclude that the given file is Suspicious. We can include more words in the search bar if required.

6. Limitations

6.1. Limitations (Previous Semester)

1. Authentic Dataset - We are yet to get our dataset from LDCIL Mysore after a few formalities. We thus trained our models on the Google Synth Dataset (comprising 38 hours of data of 2 female and 2 male synthetic speakers) that we found on the internet. The training can be better for an even more extensive data.

2. As discussed previously, the Whisper AI being an extensive model requires a good enough GPU. Also, the further rigorous training of Wav2Vec too requires a GPU more than the one provided by Google Colab (15 GB). In addition, the runtime of the colab GPU is also less and gets exhausted very quickly.

3. The GUI is quite basic. Better GUI can be made with the help of libraries Streamlit and PyQt. However, these are not supported on Google Colab.

4. We tried sentiment analysis as well, however, it didn't provide good results in Punjabi Language (the model wasn't able to differentiate between normal and suspicious words). We aim to work on it in the next semester.

6.2. Solutions to the Limitations in the previous semester

1. We successfully got the authentic LDCIL dataset. It is not synthetic unlike the Google Synth Dataset, thereby, making the model more robust and better in real world applications.

2. We got GPU access from the IT section of IIT Ropar. Due to this GPU, we were able to train our Speech-to-text models either entirely (on small datasets) or by transfer learning (on large datasets). The issue of disconnectivity on Google Colab was resolved by using the tmux terminal for training and testing for longer durations.

3. The GUI in colab didn’t work properly. So, now we have used the ‘tkinter’ library to make a better and easier GUI on the local system. The new GUI includes functionalities for color coding and semantic search as well.
4. We discarded those semantic models which didn't work properly on the LDCIL dataset.

6.3. Limitations (This Semester)

- The whisper model is getting trained only on 1 epoch due to GPU container overuse (as stated by the IT section). Training on more epochs will give even better results.
- Fine Tuning of Semantic Search model on Punjabi text files for better output of Keyword and Semantic Search
- GUI runs on the local system and is not deployed to protect sensitive data and maintain information security.

7. Future Work

1. Training the speech-to-text model (Whisper AI and Wav2Vec) on larger and authentic datasets provided by LDCIL, Mysore in an iterative manner (save and train after every epoch) considering the GPU memory limitations. One can access the GPU under the name of the concerned professor or M.Tech/PhD(s) involved in the project for better memory allocation.

2. Training on Augmented datasets, it contains augmentations - loudness, high pass filter, gaussian noise, increased speed inorder to make the model robust.

3. Fine tune the sentiment analysis model for better results.

4. Replace words from predicted transcripts with the approximate matches before running keyword search - Keyword Search

5. Developing a GUI with React instead of Python as it’s generally preferred due to limitations on access of desired python libraries and the ease of React GUIs to be extended on mobile applications.

6. Discuss the exact deliverables with the concerned authorities and try to implement possible changes/ add ons thereafter.

8. Summary

We began by implementing speech-to-text conversion using both Whisper AI and Wav2Vec models, intending to determine the optimal model based on performance. The choice of the final model will depend on achieving higher accuracy, measured by lower Word Error Rates (WER), when trained on larger, high-quality datasets using a robust GPU. Based on that, we are now moving ahead with Whisper AI for Speech-to-Text conversion due to its considerably low WER in comparison to Wav2Vec. Instead of training the entire Whisper AI model, we just trained its decoder, since it has a relatively lesser number of parameters, in context to memory issues. After obtaining the transcripts, we integrated a GUI that supports keyword-based search, enabling users to locate exact or related terms within the transcripts efficiently. We have color coded the exact, related and synonymous words. Moving beyond keyword matching, we also explored semantic search capabilities. This approach focuses on understanding the context and intent behind user queries to deliver more relevant results, even when the exact words are not present. All these keyword searches can be used to check if a given transcript file is suspicious.

9. References

[1] - Supervised vs. Unsupervised Learning: What's the Difference? | IBM

[2] - What Is Self-Supervised Learning? | IBM

[3] - OPEN AI | All About Whisper

[4] - DrishtiSharma/whisper-large-v2-punjabi-700-steps. Hugging Face

[5] - Sequence Modeling with CTC

[6] - manandey/wav2vec2-large-xlsr-punjabi · Hugging Face

[7] - Semantic Search with FAISS

[8] - Getting started with semantic search

[9] - Introducing the Beginner's Guide to Text Embeddings

[10] - sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2

[11] - Jupyter Notebook - IPyWidgets.

[12] - KEYWORD_EXTRACTION_FOR_PUNJABI_LAGUAGE (Indian Journal of Computer Science and Engineering)

[13]sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2 Hugging Face

[14]Punjabi Alphabetical Dictionary Dataset