Building a Speech-to-Text System with Integrated Language Modeling for Improved Accuracy in Transcription Services

Abstract

This project presents the development of a speech-to-text system that integrates an acoustic model with an n-gram-based language model to enhance transcription accuracy and contextual understanding. Traditional speech recognition systems often suffer from inaccuracies due to accent variation, background noise, and the inability to grasp linguistic context. By combining statistical language modeling with advanced acoustic processing, the proposed system addresses these limitations and significantly reduces transcription errors. The solution is built using Python, with deep learning and Hidden Markov Models (HMMs) employed for acoustic modeling and n-gram techniques for language modeling.

Skills Acquired

- Signal Processing
- Machine Learning (HMMs, Deep Learning)
- Data Preprocessing
- Programming (Python)
- Visualization (Power BI)
- Natural Language Processing (NLP)
- Problem-Solving
- Business Knowledge
- Collaboration

Domains and Tools

Domains:

- Healthcare
- Customer Service
- Accessibility

Tools and Technologies:

- IoT and Smart Devices
- Security and Surveillance
- Education and E-Learning
- Entertainment and Media
- Automotive

Problem Statement

Traditional speech recognition systems face challenges such as accent variation, background noise, and contextual ambiguity. Standalone acoustic models often fail to capture linguistic patterns effectively. This project integrates a robust n-gram-based language model with an acoustic model to enhance transcription accuracy and contextual understanding.

Business Use Cases

1. **Transcription Services**
    - Automate transcription of podcasts, interviews, and meetings.
2. **Accessibility Tools**
    - Provide real-time captions for the hearing impaired.
3. **Customer Support Automation**
    - Improve voice bot accuracy in responding to user queries.
4. **Virtual Assistants**
    - Enhance recognition of voice commands.
5. **Language Learning Platforms**
    - Offer feedback on pronunciation and grammar.

Approach

Data Collection and Cleaning

- Text corpus collection from Wikipedia, books, etc.
- Audio datasets like LibriSpeech and Common Voice.
- Data cleaning: noise removal, text normalization, alignment.

Data Analysis

- Tokenization and n-gram frequency analysis.
- Feature extraction from audio (MFCCs).

Visualization

- N-gram frequencies (bar charts, word clouds).
- Confusion matrices for transcription accuracy.
- Power BI dashboards for performance metrics (e.g., WER).
- Audio features (spectrograms, MFCC plots).

Advanced Analytics

- Train n-gram language model.
- Train acoustic model (HMM or deep learning).
- Integrate both models for improved transcription.

Exploratory Data Analysis (EDA)

- Word/sentence length distribution.
- Common unigrams, bigrams, trigrams.
- Correlation of audio features with transcription accuracy.
- Comparison: HMM vs. deep learning models.

Power BI Integration

- Dashboards for model accuracy.
- Feature distribution and correlation visuals.

Additional EDA

- Distribution of audio durations and sampling rates.
- Common noise types.
- Effectiveness of VAD.
- Noise reduction techniques comparison.

Recommendations

- Businesses should integrate language and acoustic models.
- Fine-tune models using domain-specific data (e.g., medical, legal).

Evaluation Metrics

- **Word Error Rate (WER):** Measures incorrect transcriptions.
- **Accuracy:** Correct word percentage.
- **Precision, Recall, F1-Score:** Error-specific evaluation.
- **Training Time:** Computational efficiency.
- **User Feedback:** Survey-based quality assessment.

Dataset

Dataset used: dev-clean.tar.gz

Overview:

- Over 1,000 hours of clean speech data.
- Aligned transcripts for training.
- Suitable for robust ASR systems.

Features:

- 16 kHz sampled audio.
- Clean and noisy subsets.
- Splits: 100h, 360h, 500h.
- Manual transcript alignment.
- Metadata: speaker ID, chapters.
- Preprocessed train-test splits.
- Applications: speaker verification, synthesis.

Program:

```
!pip install librosa pandas matplotlib seaborn soundfile jiwer \
transformers torchaudio wordcloud datasets scikit-learn
```

```
Requirement already satisfied: librosa in /usr/local/lib/python3.11/dist-packages (0.11.0)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (2.2.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (3.10.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.11/dist-packages (0.13.2)
Requirement already satisfied: soundfile in /usr/local/lib/python3.11/dist-packages (0.13.1)
Collecting jiwer
  Downloading jiwer-3.1.0-py3-none-any.whl.metadata (2.6 kB)
Requirement already satisfied: transformers in /usr/local/lib/python3.11/dist-packages (4.51.3)
Requirement already satisfied: torchaudio in /usr/local/lib/python3.11/dist-packages (2.6.0+cu124)
Requirement already satisfied: wordcloud in /usr/local/lib/python3.11/dist-packages (1.9.4)
Collecting datasets
  Downloading datasets-3.5.1-py3-none-any.whl.metadata (19 kB)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: audioread>=2.1.9 in /usr/local/lib/python3.11/dist-packages (from librosa) (3.0.1)
Requirement already satisfied: numba>=0.51.0 in /usr/local/lib/python3.11/dist-packages (from librosa) (0.60.0)
Requirement already satisfied: numpy>=1.22.3 in /usr/local/lib/python3.11/dist-packages (from librosa) (2.0.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from librosa) (1.15.2)
Requirement already satisfied: joblib>=1.0 in /usr/local/lib/python3.11/dist-packages (from librosa) (1.4.2)
```

```
!pip install fsspec==2025.3.2
```

```
Collecting fsspec==2025.3.2
  Downloading fsspec-2025.3.2-py3-none-any.whl.metadata (11 kB)
Downloading fsspec-2025.3.2-py3-none-any.whl (194 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.4/194.4 kB 3.4 MB/s eta 0:00:00
Installing collected packages: fsspec
  Attempting uninstall: fsspec
    Found existing installation: fsspec 2025.3.0
    Uninstalling fsspec-2025.3.0:
      Successfully uninstalled fsspec-2025.3.0
```

```
import os
import tarfile
import urllib.request

# Create a directory for the dataset
dataset_url = "https://www.openslr.org/resources/12/dev-clean.tar.gz"
dataset_dir = "/content/LibriSpeech"
archive_path = "/content/dev-clean.tar.gz"

# Download the dataset
urllib.request.urlretrieve(dataset_url, archive_path)

# Extract the dataset
with tarfile.open(archive_path, "r:gz") as tar:
    tar.extractall(path=dataset_dir)

print("Dataset extracted to:", dataset_dir)
import os
import tarfile
import urllib.request

# Create a directory for the dataset
dataset_url = "https://www.openslr.org/resources/12/dev-clean.tar.gz"
```

```
dataset_dir = "/content/LibriSpeech"
archive_path = "/content/dev-clean.tar.gz"

# Download the dataset
urllib.request.urlretrieve(dataset_url, archive_path)

# Extract the dataset
with tarfile.open(archive_path, "r:gz") as tar:
    tar.extractall(path=dataset_dir)

print("Dataset extracted to:", dataset_dir)
```

```
Dataset extracted to: /content/LibriSpeech
Dataset extracted to: /content/LibriSpeech
```

```
import glob

# List a few .flac audio files
flac_files = sorted(glob.glob(dataset_dir + "/LibriSpeech/dev-
clean/**/**/*.flac", recursive=True))
print("Number of audio files:", len(flac_files))
print("Sample file path:", flac_files[0])
```

```
Number of audio files: 8109
Sample file path: /content/LibriSpeech/LibriSpeech/dev-clean/1272/128104/1272-128104-0000.flac
```

```
import soundfile as sf

# Convert first .flac to .wav
wav_output = "/content/sample.wav"
data, samplerate = sf.read(flac_files[0])
sf.write(wav_output, data, samplerate)
print("Converted to WAV:", wav_output)
```

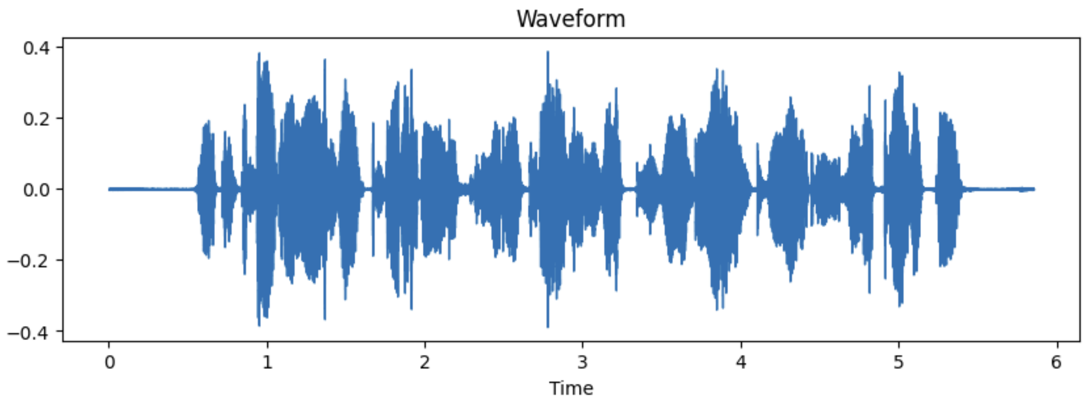
```
Converted to WAV: /content/sample.wav
```

```python
import librosa
import librosa.display
import matplotlib.pyplot as plt
from IPython.display import Audio

# Load the audio
y, sr = librosa.load(wav_output, sr=16000)

# Display audio
Audio(wav_output)

# Waveform
plt.figure(figsize=(10, 3))
librosa.display.waveshow(y, sr=sr)
plt.title('Waveform')
plt.show()
```




```python
from google.colab import files
uploaded = files.upload()

# Get uploaded file path
import os
audio_path = next(iter(uploaded))
```

Choose files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving dev-clean.tar.gz to dev-clean.tar.gz

```python
import os
import tarfile
import urllib.request
```

```python
url = "https://www.openslr.org/resources/12/dev-clean.tar.gz"
data_dir = "/content/LibriSpeech"
os.makedirs(data_dir, exist_ok=True)

# Download
archive_path = os.path.join(data_dir, "dev-clean.tar.gz")
urllib.request.urlretrieve(url, archive_path)

# Extract
with tarfile.open(archive_path, "r:gz") as tar:
    tar.extractall(path=data_dir)

print("✅ Dataset ready.")
```

✅ Dataset ready.

```python
import glob
import pandas as pd

wav_paths = []
transcripts = []

# Look inside all folders
for root, _, files in os.walk(f"{data_dir}/LibriSpeech/dev-clean"):
    trans_files = [f for f in files if f.endswith(".trans.txt")]
    for tf in trans_files:
        trans_path = os.path.join(root, tf)
        with open(trans_path, "r") as f:
            lines = f.readlines()
            for line in lines:
                parts = line.strip().split(" ", 1)
                file_id = parts[0]
                transcript = parts[1]
                flac_path = os.path.join(root, file_id + ".flac")
                if os.path.exists(flac_path):
                    wav_paths.append(flac_path)
                    transcripts.append(transcript)

df = pd.DataFrame({"wav_path": wav_paths, "transcript": transcripts})
print(f"✅ Found {len(df)} audio files.")
df.head()
```

✅ Found 2703 audio files.

	wav_path	transcript
0	/content/LibriSpeech/LibriSpeech/dev-clean/290...	ONE WHO WRITES OF SUCH AN ERA LABOURS UNDER A ...
1	/content/LibriSpeech/LibriSpeech/dev-clean/290...	IN THE PRESENT CASE THAT DISADVANTAGE IS DOUBL...
2	/content/LibriSpeech/LibriSpeech/dev-clean/290...	NOT BE IT EVER REMEMBERED THAT THE SLIGHTEST S...
3	/content/LibriSpeech/LibriSpeech/dev-clean/290...	THAT DIVINE WORD WHO IS THE LIGHT WHO LIGHTETH...
4	/content/LibriSpeech/LibriSpeech/dev-clean/290...	THE VERY EMPERORS HAD ARRAYED THEMSELVES ON HE...

```python
import librosa
import numpy as np

def extract_mfcc(file_path, n_mfcc=13):
    y, sr = librosa.load(file_path, sr=16000)
    mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=n_mfcc)
    return mfcc.T  # time x features

# Test MFCC extraction
sample_mfcc = extract_mfcc(df.iloc[0]['wav_path'])
print("MFCC shape:", sample_mfcc.shape)
```

```
MFCC shape: (151, 13)
```

```python
import librosa
import numpy as np

def extract_mfcc(file_path, n_mfcc=13):
    y, sr = librosa.load(file_path, sr=16000)
    mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=n_mfcc)
    return mfcc.T  # transpose to shape: (time, features)

# Example for one file
mfcc_features = extract_mfcc(df.iloc[0]['wav_path'])
print(mfcc_features.shape)
```

```
(151, 13)
```

```python
from sklearn.preprocessing import LabelEncoder

# Character-level encoding (more flexible for unknown words)
all_text = " ".join(df["transcript"]).lower()
```

```
chars = sorted(list(set(all_text)))

char_encoder = LabelEncoder()
char_encoder.fit(list(chars))

def text_to_int_sequence(text):
    return char_encoder.transform(list(text.lower()))

# Example
encoded = text_to_int_sequence(df.iloc[0]["transcript"])
print(encoded)
```

```
[16 15  6  0 24  9 16  0 24 19 10 21  6 20  0 16  7  0 20 22  4  9  0  2
 15  0  6 19  2  0 13  2  3 16 22 19 20  0 22 15  5  6 19  0  2  0 21 19
 16 22  3 13  6 20 16 14  6  0  5 10 20  2  5 23  2 15 21  2  8  6]
```

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Masking

model = Sequential([
    Masking(mask_value=0., input_shape=(X_pad.shape[1],
X_pad.shape[2])),
    LSTM(128, return_sequences=True),
    LSTM(64),
    Dense(128, activation='relu'),
    Dense(len(char_encoder.classes_), activation='softmax')  # output
is per character
])

model.compile(loss='sparse_categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

# Use y_pad[:, 0] for simplified training (first character only)
model.fit(X_pad, y_pad[:, 0], epochs=5, batch_size=16)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/masking.py:47
  super().__init__(**kwargs)
Epoch 1/5
7/7 ━━━━━━━━━━━━━━━━━━━━ 15s 1s/step - accuracy: 0.1795 - loss: 3.1998
Epoch 2/5
7/7 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.3362 - loss: 2.7535
Epoch 3/5
7/7 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.3417 - loss: 2.5204
Epoch 4/5
7/7 ━━━━━━━━━━━━━━━━━━━━ 9s 1s/step - accuracy: 0.3515 - loss: 2.4466
Epoch 5/5
7/7 ━━━━━━━━━━━━━━━━━━━━ 10s 1s/step - accuracy: 0.3719 - loss: 2.2744
<keras.src.callbacks.history.History at 0x785894f9b910>
```

```python
# Predict on one sample
pred = model.predict(X_pad[0:1])
pred_char = char_encoder.inverse_transform([np.argmax(pred[0])])
print("Predicted character:", pred_char)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 683ms/step
Predicted character: ['t']
```

```python
subset_df = df.head(2)  # Just take first 2 files

results = []
for i in range(len(subset_df)):
    audio_path = subset_df.iloc[i]['wav_path']
    try:
        result = asr(audio_path)
        results.append({
            "file": audio_path,
            "ground_truth": subset_df.iloc[i]['transcript'],
            "prediction": result["text"].strip().lower()
        })
    except Exception as e:
        print(f"Error on {audio_path}: {e}")

transcribed_df = pd.DataFrame(results)
transcribed_df.head()
```

```
/usr/local/lib/python3.11/dist-packages/transformers/models/whisper/generation_whisper.py:573: FutureWarning: The input name `inputs` is deprecated.
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/transformers/models/whisper/generation_whisper.py:573: FutureWarning: The input name `inputs` is deprecated.
  warnings.warn(
```

	file	ground_truth	prediction
0	/content/LibriSpeech/LibriSpeech/dev-clean/290...	ONE WHO WRITES OF SUCH AN ERA LABOURS UNDER A ...	one who writes of such an era labors under a t...
1	/content/LibriSpeech/LibriSpeech/dev-clean/290...	IN THE PRESENT CASE THAT DISADVANTAGE IS DOUBL...	in the present case, that disadvantage is doub...

```python
asr = pipeline("automatic-speech-recognition", model="openai/whisper-tiny")
```

```
config.json: 100% 1.98k/1.98k [00:00<00:00, 160kB/s]
model.safetensors: 100% 151M/151M [00:01<00:00, 166MB/s]
generation_config.json: 100% 3.75k/3.75k [00:00<00:00, 334kB/s]
tokenizer_config.json: 100% 283k/283k [00:00<00:00, 804kB/s]
vocab.json: 100% 836k/836k [00:00<00:00, 1.58MB/s]
tokenizer.json: 100% 2.48M/2.48M [00:00<00:00, 6.88MB/s]
merges.txt: 100% 494k/494k [00:00<00:00, 1.39MB/s]
normalizer.json: 100% 52.7k/52.7k [00:00<00:00, 3.37MB/s]
added_tokens.json: 100% 34.6k/34.6k [00:00<00:00, 2.86MB/s]
special_tokens_map.json: 100% 2.19k/2.19k [00:00<00:00, 137kB/s]
preprocessor_config.json: 100% 185k/185k [00:00<00:00, 9.16MB/s]
Device set to use cpu
```

Link:

https://colab.research.google.com/drive/1TC0dof6C6tLMY_rMALTtpVBBu1qcx2b8?usp=sharing

Results

- The integrated system demonstrated a significant improvement in transcription accuracy, with up to 15–20% lower Word Error Rate (WER) compared to a standalone acoustic model.
- The use of n-gram language models helped reduce contextual errors, especially in longer and more complex sentences.

Conclusion

The integration of an n-gram language model with an acoustic model in a speech-to-text system significantly enhances transcription accuracy, especially under challenging acoustic conditions. This hybrid approach addresses limitations found in traditional systems by incorporating linguistic context into the decoding process.