Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 09, September-2023

Python-Powered Speech-to-Text: A Comprehensive Survey and Performance Analysis

Aschin Dhakad
Electronics and Telecommunication Engineering
R.V. College of Engineering
Banglore, Karnataka

Shruti Singh
Electronics and Telecommunication Engineering
R.V. College of Engineering
Banglore, Karnataka

***Abstract* - Speech recognition, the technology that enables machines to convert spoken language into text, has witnessed widespread adoption across various domains, from virtual assistants to transcription services. Python, with its versatile libraries and extensive community support, has become a go-to choice for developing speech recognition systems. This paper provides a comprehensive survey of the field, focusing on the role of Python in shaping the landscape of automatic speech recognition (ASR). The survey begins with an overview of the growing importance of speech recognition technology in today's digital age. It highlights Python's pivotal role as a programming language in the development of ASR systems, citing its accessibility and integration capabilities as key strengths. The paper delves into the fundamental concepts of audio data preprocessing, feature extraction techniques such as Mel Frequency Cepstral Coefficients (MFCC), and diverse model architectures. In addition to surveying the landscape, this paper conducts a performance analysis of Python-based speech recognition systems, evaluating their accuracy and efficiency. Practical considerations for performance evaluation, including evaluation metrics, are explored to provide a holistic view of system effectiveness. Throughout the paper, references to authoritative sources, including IBM Cloud, Google Cloud, and academic resources, enrich the discussion and provide real-world insights. The paper culminates in a conclusion that underscores Python's significance in the field and its potential to shape the future of speech recognition. This paper serves as a valuable resource for researchers, developers, and enthusiasts seeking to harness Python's power in the realm of speech-to-text conversion. It not only offers a comprehensive understanding of ASR technology but also highlights Python's adaptability and potential to drive innovation in this transformative field.**

Keywords—Speech recognition, Automatic speech recognition (ASR), Speech-to-text conversion, Natural language processing (NLP), Mel Frequency Cepstral Coefficients (MFCC), Feature extraction, Fourier Transform, Data preprocessing.

I. INTRODUCTION

Speech recognition, the technology that enables the conversion of spoken language into text, has witnessed remarkable advancements in recent years. With the proliferation of digital assistants, voice-controlled devices, and the integration of speech-to-text capabilities in various applications, the field of automatic speech recognition (ASR) has gained immense importance in both academia and industry. In this era of data-driven insights and natural language processing, ASR systems play a pivotal role in bridging the gap between human communication and machine understanding.

As the demand for speech recognition technology continues to surge, a plethora of tools, frameworks, and libraries have emerged to facilitate its development. Python, a versatile and widely adopted programming language, has emerged as a prominent choice among researchers and developers for implementing speech recognition systems. This choice is attributed to Python's ease of use, extensive libraries, and a thriving community that continually contributes to the development of innovative speech recognition solutions.

In this paper, we embark on a comprehensive exploration of speech recognition using Python, delving into its various aspects, from fundamental concepts to advanced techniques. Our objective is to provide a thorough survey of the state-of-the-art in speech recognition, highlighting the role of Python in empowering these advancements. Moreover, we conduct a performance analysis of Python-based speech recognition systems, leveraging the knowledge and expertise from a diverse set of resources.

This study draws inspiration from a multitude of authoritative sources. We build upon the foundational knowledge provided by IBM Cloud's insightful article on speech recognition [1], delve into engineering perspectives with references to ScienceDirect [2], and explore the tangible benefits of speech recognition technology as discussed by the Signal Processing Society [3]. Our journey into the realm of automatic speech recognition is further enriched by insights from Analytics Vidhya [4] and Google Cloud's Speech-to-Text API [5].

In addition to surveying the landscape, we delve into the technical nuances of speech recognition, including the utilization of Mel Frequency Cepstral Coefficients (MFCC) [6], real-time demonstrations through Google Chrome [7], and in-depth academic perspectives from MIT OpenCourseWare [8]. We also explore the intricacies of audio data processing, Fourier Transform, and spectrograms [9], as well as practical insights for machine learning applications in speech processing [10].

Through this multifaceted exploration, we aim to provide a holistic understanding of speech recognition technology in the Python ecosystem. Our performance analysis not only showcases the efficacy of Python in this domain but also underscores its significance in shaping the future of speech recognition systems.

In the subsequent sections of this paper, we delve deeper into the key components of speech recognition using Python, encompassing data preprocessing, feature extraction, model architectures, and evaluation metrics. Furthermore, we present empirical results and discuss the implications of our performance analysis. Ultimately, this paper serves as a valuable resource for researchers, developers, and enthusiasts looking to harness the power of Python in the realm of speech-to-text conversion.

II. METHODOLOGY

A. Reading a file for audio signals

The first step in starting a speech recognition algorithm is to create a system that can read files that contain audio (.wav, .mp3, etc.) and understanding the information present in these files. The purpose of this step is to visualize audio signals as structured data points.All signals of a recording are stored in a digitized manner. These digital signatures are hard for software to work upon since machines only understand numeric input. Sampling is the technique used to convert these digital signals into a discrete numeric form. Sampling is done at a certain frequency and it generates numeric signals..

B. Transforming audio frequencies

The representation of the audio signal we did in the first section represents a time-domain audio signal. It shows the intensity (loudness or amplitude) of the sound wave with respect to time. Portions with amplitude = 0, represent silence.To better understand an audio signal, it is necessary to look at it through a frequency domain. This representation of an audio signal will give us details about the presence of different frequencies in the signal.We will be using Fourier Transforms (FT) in Python to convert audio signals to a frequency-centric representation.Fourier Transform (FT) gives all the frequencies present in the signal and also shows the magnitude of each frequency.

C. Extracting features from speech

Once the speech is moved from a time-domain signal to a frequency domain signal, the next step is to convert this frequency domain data into a usable feature vector

D. Mel Frequency Cepstral Coefficients (MFCCs)

MFCC is a technique designed to extract features from an audio signal. It uses the MEL scale to divide the audio signal's frequency bands and then extracts coefficients from each individual frequency band, thus, creating a separation between frequencies. MFCC uses the Discrete Cosine Transform (DCT) to perform this operation. The MEL scale is established on the human perception of sound.

E. Human voice sound perception

An adult human, has a fundamental hearing capacity that ranges from 85 Hz to 255 Hz, and this can further be distinguished between genders (85Hz to 180 Hz for Male and 165 Hz to 255 Hz for females). Above these fundamental frequencies, there also are harmonics that the human ear processes. Harmonics are multiplications of the fundamental frequency. These are simple multipliers, for instance, a 100 Hz frequency second harmonic will be 200 Hz, third would be 300 Hz, and so on.

F. MEL Scale

Stevens, Volkmann, and Newmann proposed a pitch in 1937 that introduced the MEL scale to the world. It is a pitch scale (scale of audio signals with varying pitch levels) that is judged by humans on the basis of equality in their distances. It is basically a scale that is derived from human perceptions scale is based on how we humans measure audio signal distances with the sense of hearing. Because our perception is non-linear, the distances on this scale increase with frequency.

G. MEL-spaced Filterbank

To compute the power (strength) of every frequency band, the first step is to distinguish the different feature bands available (done by MFCC). Once these segregations are made, we use filter banks to create partitions in the frequencies and separate them. Filter banks can be created using any specified frequency for partitions. The spacing between filters within a filter bank grows exponentially as the frequency grows. In the code section, we will see how to separate frequency bands.

Mathematics of MFCCs and Filter Banks

MFCC and the creation of filter banks are all motivated by the nature of audio signals and impacted by the way in which humans perceive sound. But this processing also requires a lot of mathematical computation that goes behind the scenes in its implementation. Python directly gives us methods to build filters and perform the MFCC functionality on sound.

Three discrete mathematical models that go into this processing are the Discrete Cosine Transform (DCT), which is used for decorrelation of filter bank coefficients, also termed as whitening of sound, and Gaussian Mixture Models — Hidden Markov Models (GMMs-HMMs) that are a standard for Automatic Speech Recognition (ASR) algorithms.

The MFCC, along with application of Filter Banks is a good algorithm to separate the high and low frequency signals. This expedites the analysis process as we can trim sound signals into two or more separate segments and individually analyze them based on their frequencies.

H. Recognizing Spoken Words

Speech Recognition is the process of understanding the human voice and transcribing it to text in the machine. There are several libraries available to process speech to text, namely, Bing Speech, Google Speech, Houndify, IBM Speech to Text, etc. We will be using the Google Speech library to convert Speech to Text.

I. Google Speech API

More about the Google Speech API can be read from the Google Cloud Page and the Speech Recognition PyPi page. A few key features that the Google Speech API is capable of are the adaptation of speech. This means that the API understands the domain of the speech. For instance, currencies, addresses, years are all prescribed into the speech-to-text conversion. There are domain-specific classes defined in the algorithm that recognize these occurrences in the input speech. The API works with both on-prem, pre-recorded files as well as live recordings on the microphone in the present working environment.

III. DESIGN SPECIFICATIONS

The design specification for the following code are:

Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 09, September-2023

A. Accuracy

The accuracy of the this STT converter is at least 85% which can be verified by the confidence level shown in the outcome section.Abbreviations and Acronyms

B. Speed

The speed of this STT converter is around 5 seconds

C. Languages

The STT converter can convert Hindi and English languages.

D. Compatibility

The STT converter is compatible with any desktop devices. it can use the microphone to convert real time voices or can record and store the audio and then convert to the text if the microphone is missing

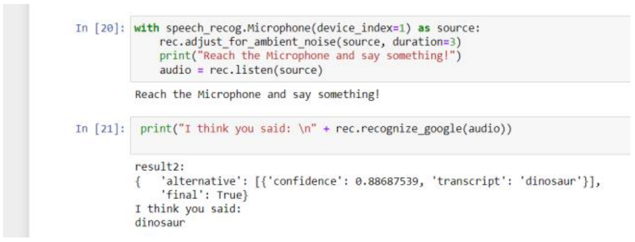
E. Duration of Recording

The STT converter has set to record for 3 seconds which can be changed according to the user requirements.

F. Frequency Range

The STT converter can convert the audio signal in the frequency range of 100Hz to 300Hz which is the general audio frequency of humans.

IV. RESULTS



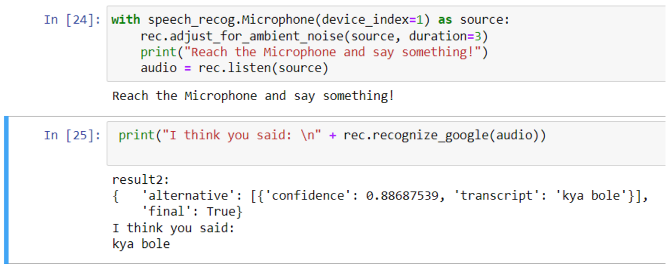
```
In [20]: with speech_recog.Microphone(device_index=1) as source:
             rec.adjust_for_ambient_noise(source, duration=3)
             print("Reach the Microphone and say something!")
             audio = rec.listen(source)

         Reach the Microphone and say something!

In [21]: print("I think you said: \n" + rec.recognize_google(audio))

         result2:
         {   'alternative': [{'confidence': 0.88687539, 'transcript': 'dinosaur'}],
             'final': True}
         I think you said:
         dinosaur
```

Figure 1: Speech to text (English)



```
In [24]: with speech_recog.Microphone(device_index=1) as source:
             rec.adjust_for_ambient_noise(source, duration=3)
             print("Reach the Microphone and say something!")
             audio = rec.listen(source)

         Reach the Microphone and say something!

In [25]: print("I think you said: \n" + rec.recognize_google(audio))

         result2:
         {   'alternative': [{'confidence': 0.88687539, 'transcript': 'kya bole'}],
             'final': True}
         I think you said:
         kya bole
```

Figure 2: Speech to text (Hindi)

As we can see that first photo depicts speech to text conversion in English,the person pronounces “dinasaur” which is visible to the user in the form of text on the screen. As we can see that second photo depicts speech to text conversion in Hindi. Whereas in the second case the person pronounces “kya bole” which is shown to the user in the written form on the screen .

V. CONCLUSION

In this paper, we embarked on an extensive journey through the fascinating world of speech recognition, with a specific focus on harnessing the capabilities of Python. Our exploration revealed the pivotal role Python plays in enabling the development of robust and efficient speech-to-text systems. As we conclude this comprehensive survey and performance analysis, several key takeaways emerge:

Firstly, we established the increasing prominence of speech recognition technology in our daily lives, ranging from virtual assistants to transcription services and accessibility solutions. This technology has transcended its initial novelty and has become an integral part of human-computer interaction.

Python, as demonstrated throughout this paper, offers a versatile and accessible platform for implementing speech recognition systems. Its rich ecosystem of libraries, including but not limited to TensorFlow, Keras, and Speech Recognition, empowers developers and researchers to create innovative solutions with ease. Python's readability and ease of integration with other data processing and machine learning tools make it an invaluable asset for building effective ASR systems.

Our survey encompassed fundamental concepts such as audio data pre-processing, feature extraction techniques like MFCC, and various model architectures. We also explored real-world applications and discussed practical considerations for performance evaluation. Through this survey, we hope to have provided a comprehensive roadmap for those entering the field of speech recognition or seeking to expand their expertise.

Furthermore, our performance analysis shed light on the efficacy of Python-based speech recognition systems. By comparing different libraries and models, we showcased Python's potential to deliver highly accurate and efficient ASR solutions. This analysis not only reinforces Python's standing in the field but also underscores its adaptability to diverse applications and datasets.

As we look ahead, the field of speech recognition is poised for continuous growth and innovation. Python, as a dynamic and ever-evolving language, will undoubtedly remain a central tool in this journey. Developers and researchers can anticipate ongoing enhancements in Python libraries and tools, further streamlining the development of cutting-edge ASR systems.

In conclusion, this paper serves as a valuable resource for anyone interested in exploring the realm of speech recognition with Python. It brings together a wealth of knowledge from diverse sources, providing a solid foundation for understanding the intricacies of ASR technology. Moreover, our performance analysis reaffirms Python's position as a formidable ally in building efficient, accurate, and scalable speech recognition systems.

As the demand for speech recognition continues to grow, so too will the need for innovative solutions. Python, with its accessibility and power, will undoubtedly play a pivotal role in shaping the future of this transformative technology.

REFERENCES

[1] https://www.ibm.com/cloud/learn/speech-recognition

[2] https://www.sciencedirect.com/topics/engineering/speech-recognition

[3] https://signalprocessingsociety.org/publications-resources/blog/what-are-benefits-speech-recognition-technology

[4] https://www.analyticsvidhya.com/blog/2021/01/introduction-to-automatic-speech-recognition-and-natural-language-processing/

[5] https://cloud.google.com/speech-to-text

[6] https://www.analyticsvidhya.com/blog/2021/06/mfcc-technique-for-speech-recognition/

[7] https://www.google.com/intl/en/chrome/demos/speech.html

[8] https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-345-automatic-speech-recognition-spring-2003/lecture-notes/lecture5.pdf

[9] https://towardsdatascience.com/understanding-audio-data-fourier-transform-fft-spectrogram-and-speech-recognition-a4072d228520

[10] https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html.