simplilearn | PURDUE UNIVERSITY

Natural Language Processing and Speech Recognition

Text to Speech

Learning Objectives

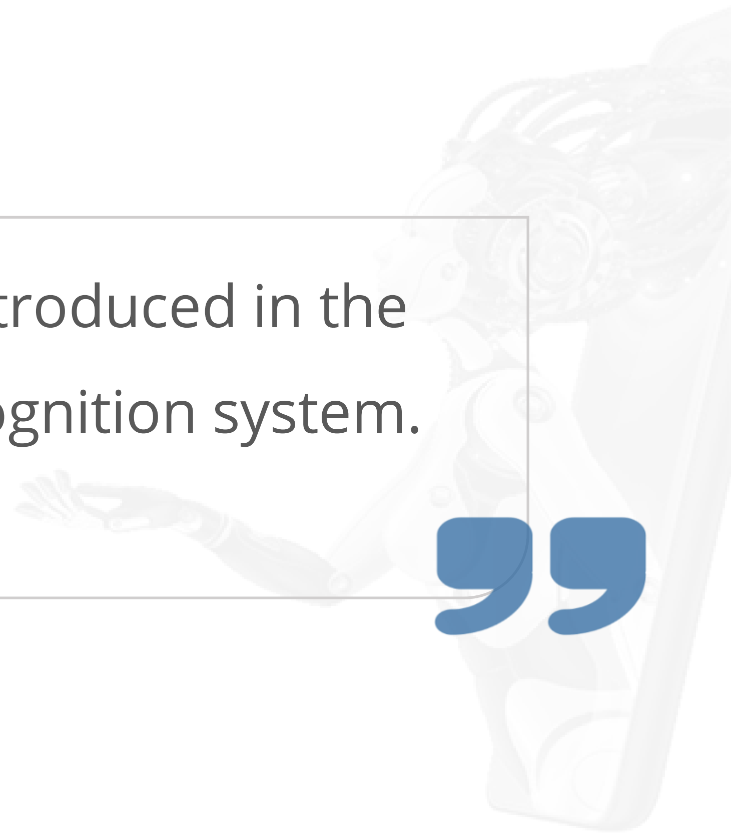
By the end of this lesson, you will be able to:

- Analyze different text data
- Explain development of speech from text
- Illustrate different python modules to convert text to speech
- Construct model to convert text to speech using python

Text to Speech Synthesizer (TTS)

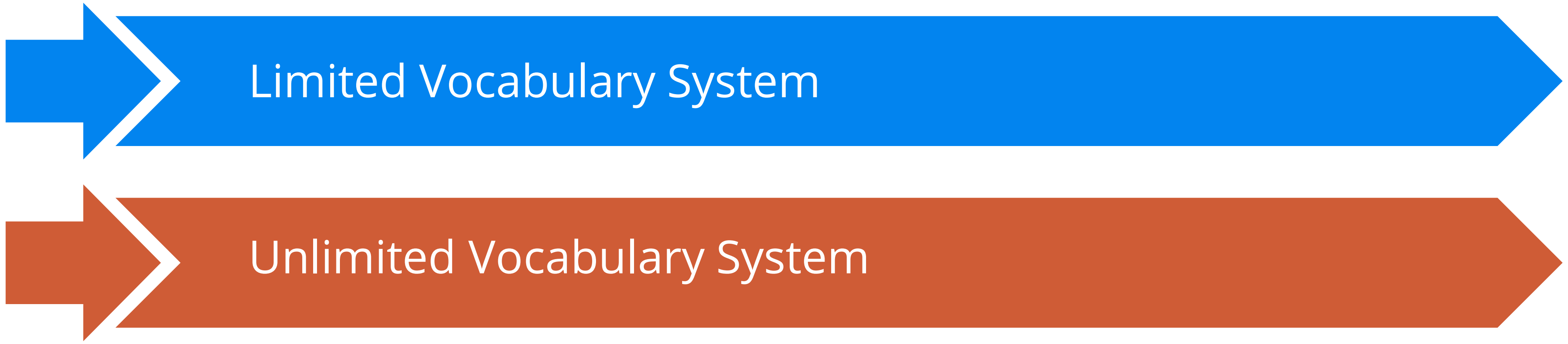
What Is TTS?

> It is a computer-based system that reads any text aloud whether it is directly introduced in the computer by an operator or scanned and submitted to an optical character recognition system.



Application of TTS

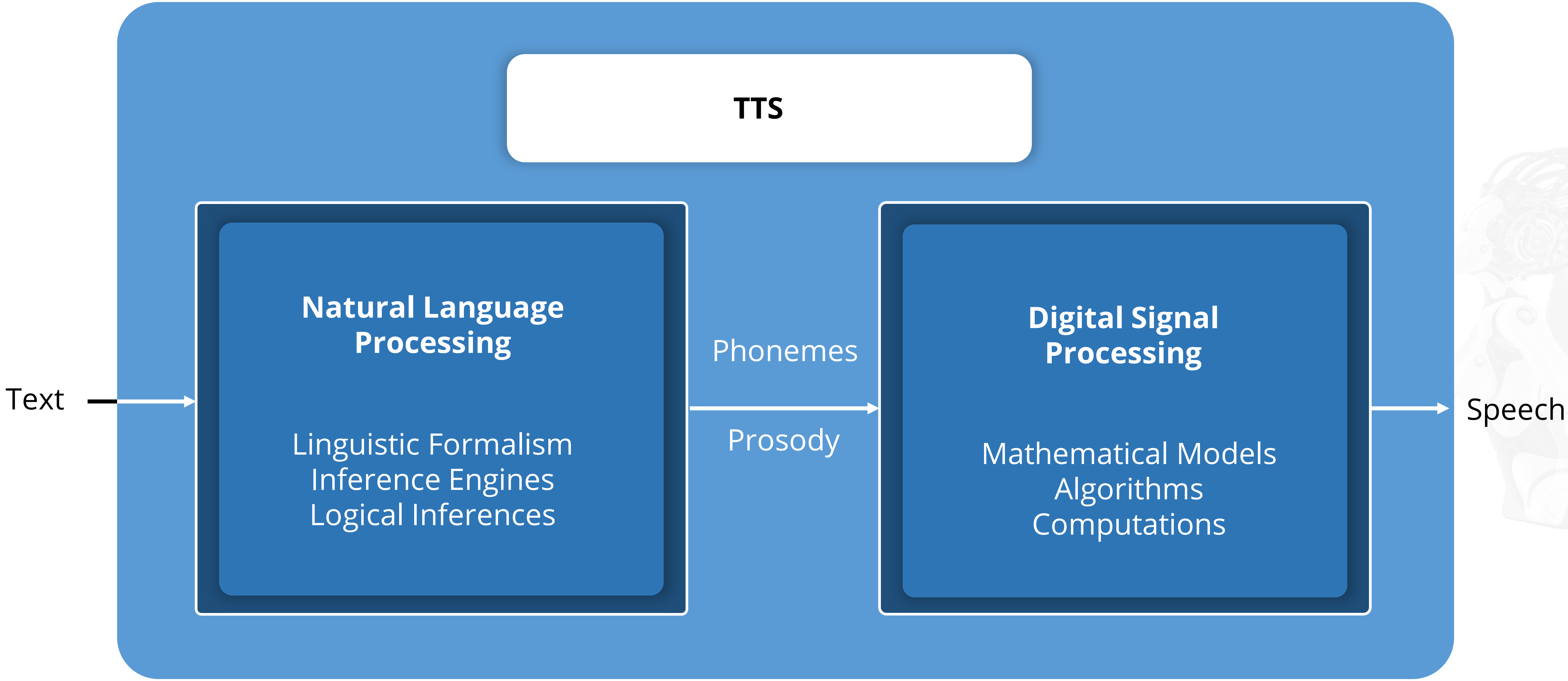
Voice response system is the application of speech synthesis technology and broadly classified into two types:



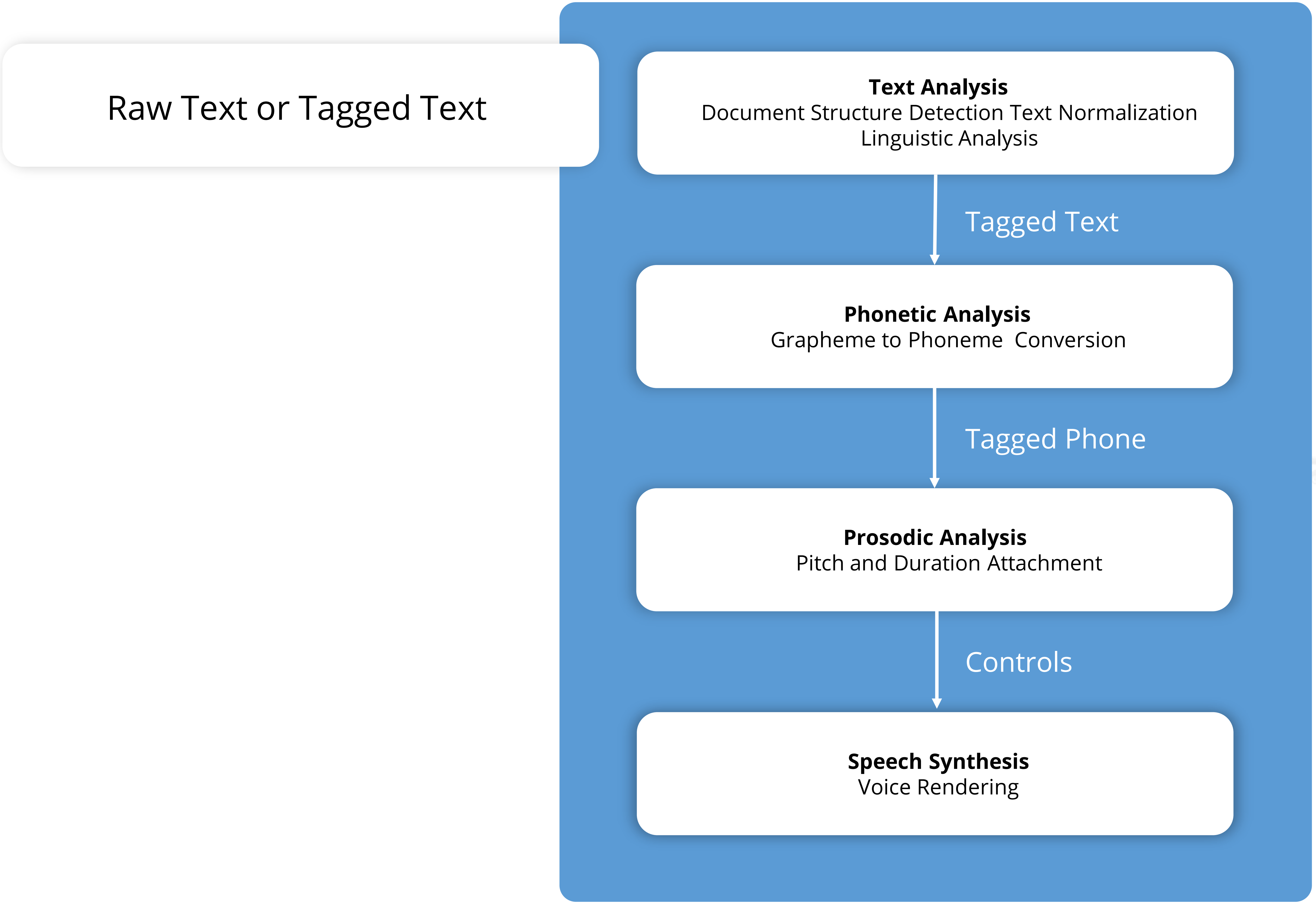
Limited Vocabulary System

Unlimited Vocabulary System

General Functional Diagram of TTS



Architecture of TTS System



Raw Text or Tagged Text

Text Analysis
Document Structure Detection Text Normalization
Linguistic Analysis

Tagged Text

Phonetic Analysis
Grapheme to Phoneme Conversion

Tagged Phone

Prosodic Analysis
Pitch and Duration Attachment

Controls

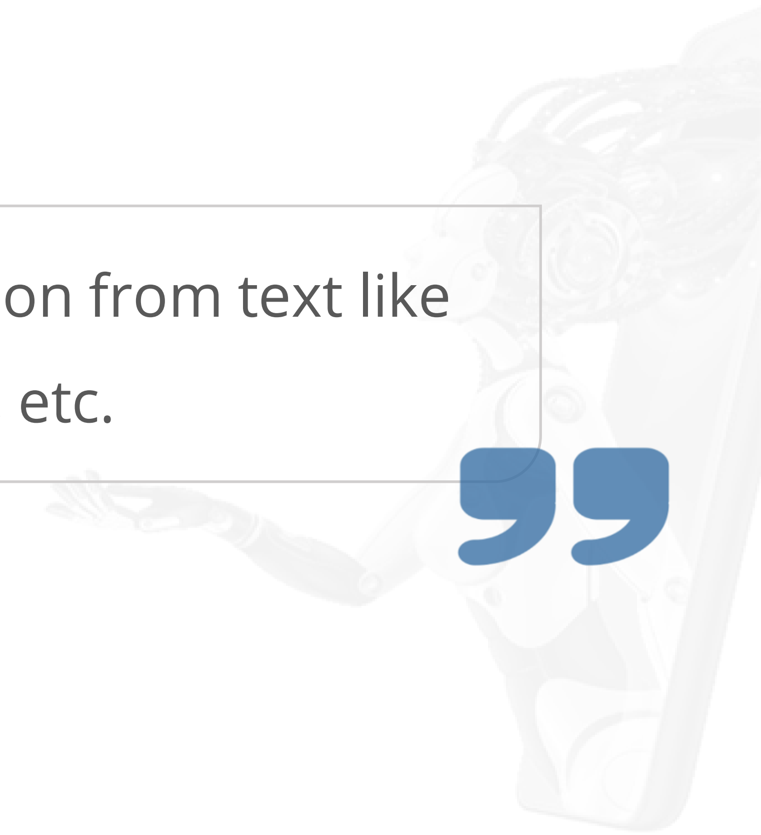
Speech Synthesis
Voice Rendering

DATA AND
ARTIFICIAL INTELLIGENCE

Text Analysis

What Is Text Analysis?

> Text analysis is a process that allows machines to extract and classify information from text like tweets, emails, support tickets, product reviews, survey responses, etc.



Methods of Text Analysis

Word Frequency

Collocation

Word Frequency

- Lists the most frequently occurring words or concepts in a given text
- Analyzes the words or expressions customers use most frequently in support conversations
- Example: The word 'delivery' appears most often, it suggests that there are issues with the company's delivery service

Collocation

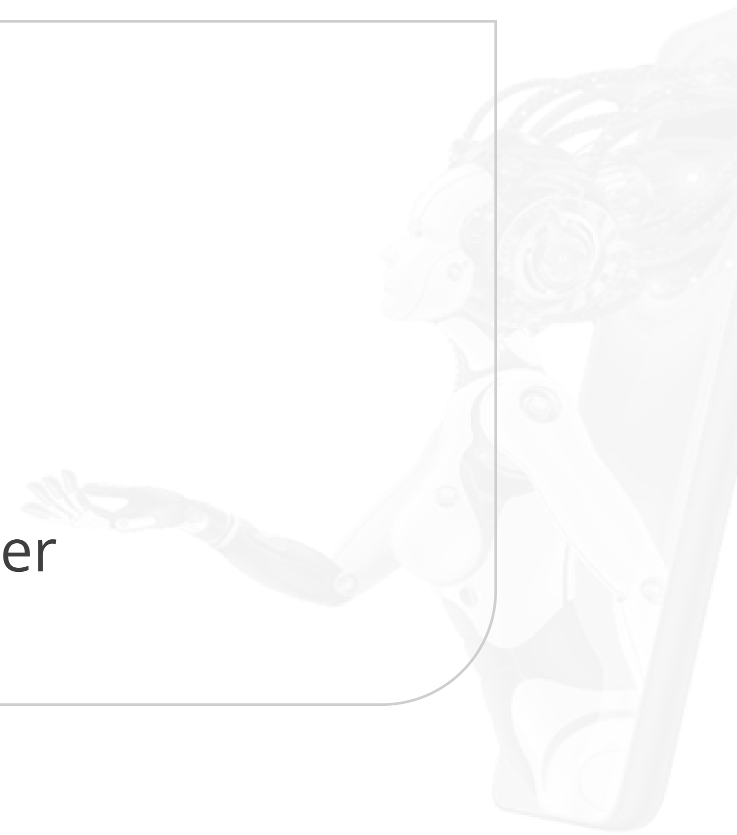
- Helps identify words that commonly co-occur
- For example, in customer reviews on a hotel booking website, the words 'air' and 'conditioning' are more likely to co-occur rather than appear individually
- Helps identify hidden semantic structures and improves the granularity of the insights by counting **bigrams** (two adjacent words, such as “air-conditioning” or “customer support”) and **trigrams** (three adjacent words, such as “out of office” or “to be continued”) as one word

Advanced Methods of Text Analysis

- Text Classification
- Sentiment Analysis

Text Classification

- Is the process of assigning predefined tags or categories to unstructured text
- Is considered one of the most useful natural language processing techniques
- Is versatile and can organize, structure, and categorize almost everything to deliver meaningful data and solve problems



Sentiment Analysis

- Is the automated process of understanding an opinion about a given subject from written or spoken language

- Comes into play where emotions are essential for effective communication between humans

Tokenization

- Is the process of breaking strings into tokens which are small structures or units
- Involves understanding the importance of each word with respect to the sentence
- Produces structural description on an input sentence

Code

```
# Importing necessary library

import pandas as pd
import numpy as np
import nltk
import os
import nltk.corpus
```

Tokenization

Code

```
# sample text for performing tokenization
text = "In Brazil they drive on the right-hand side of the road. Brazil
has a large coastline on the eastern
side of South America"

# importing word_tokenize from nltk
from nltk.tokenize import word_tokenize

# Passing the string text into word tokenize for breaking the sentences
token = word_tokenize(text)
print(token)
```

Tokenization

Output

```
Output :['In','Brazil','they','drive', 'on','the', 'right-hand', 'side',
'of', 'the', 'road', '.', 'Brazil', 'has', 'a', 'large', 'coastline',
'on', 'the', 'eastern', 'side', 'of', 'South', 'America']
```

Finding Frequency Distinct in the Text

- Import **FreqDist** library from **nltk** and pass the token into **FreqDist**

Code

```
from nltk.probability import FreqDist
fdist = FreqDist(token)
print(fdist)


OUTPUT : FreqDist({'the': 3, 'Brazil': 2, 'on': 2, 'side': 2, 'of': 2,
'In': 1, 'they': 1, 'drive': 1, 'right-hand': 1, 'road': 1, ...})
```

Finding Frequency Distinct in the Text

- Find the frequency of top 10 words

Code

```
fdist1 = fdist.most_common(10)
print(fdist1)

OUTPUT : [('the', 3),('Brazil', 2),('on', 2),('side', 2),('of', 2),('In',
1),('they', 1),('drive', 1),('right-hand', 1),('road', 1)]
```

Stemming

Stemming usually refers to normalizing words into its base form or root form.

- Import **Porterstemmer** from **nltk** library
- Check for the word "giving"

Code

```
from nltk.stem import PorterStemmer
pst = PorterStemmer()
pst.stem("waiting")
```

```
Output : 'wait'
```

Stemming

- Check for the list of words

Code

```
stm = ["waited", "waiting", "waits"]
for word in stm :
  print(word+ ":" +pst.stem(word))
```

```
Output :
waited:wait
waiting:wait
waits:wait
```

Lancaster Stemming

Lancaster stemmers are more aggressive than normal stemmers.

- Import **LancasterStemme**r from **nltk**

Code

```
from nltk.stem import LancasterStemmer
lst = LancasterStemmer()
stm = ["giving", "given", "given", "gave"]
for word in stm :
print(word+ ":" +lst.stem(word))
```

```
Output
     giving:giv
     given:giv
     given:giv
     gave:gav
```

Lemmatization

Groups together different forms of a word called lemma

Maps several words into one common root

Outputs a proper word

Considers the context of the word

Converts the word to its meaningful base form

Lemmatization

- Import **Lemmatize**r library from **nltk**

Code

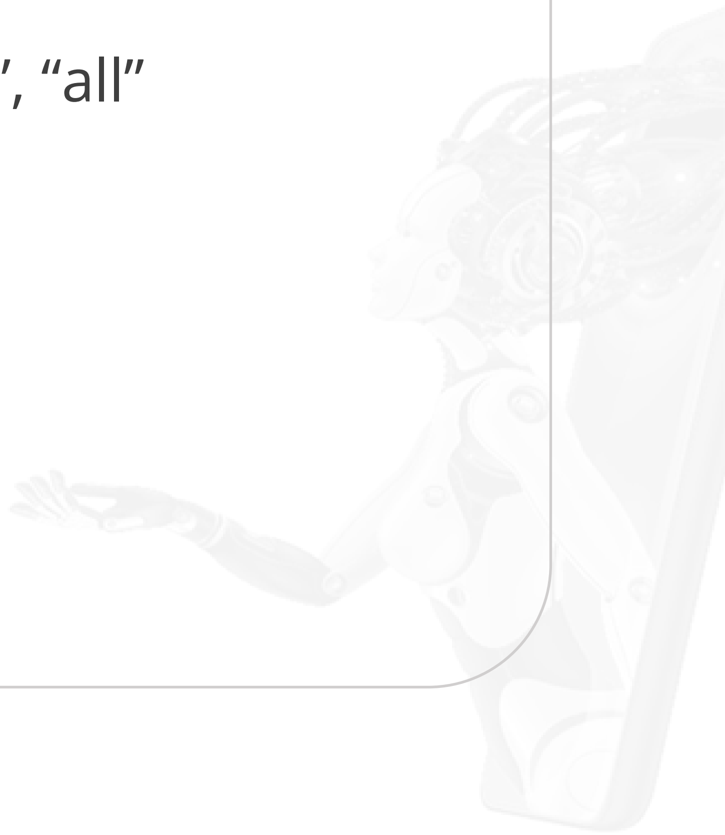
```
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()

print("rocks :", lemmatizer.lemmatize("rocks"))
print("corpora :", lemmatizer.lemmatize("corpora"))
```

```
Output :
        rocks : rock
        corpora : corpus
```

Stop Words

- Are the most common words in a language like “the”, “a”, “at”, “for”, “above”, “on”, “is”, “all”
- Do not provide any meaning and are usually removed from texts
- Can be removed using the **nltk** library



Stop Words

- Import **stopwords** from the **nltk** library

Code

```
from nltk import word_tokenize
from nltk.corpus import stopwords
a = set(stopwords.words('english'))
text = "Cristiano Ronaldo was born on February 5, 1985, in Funchal,
Madeira, Portugal."
text1 = word_tokenize(text.lower())
print(text1)
stopwords = [x for x in text1 if x not in a]
print(stopwords)
```

Stop Words

Output

```
Output of text:
['cristiano', 'ronaldo', 'was', 'born', 'on', 'february', '5', ',',
'1985', ',', 'in', 'funchal', ',', 'madeira', ',', 'portugal', '.']
Output of stopwords:
['cristiano', 'ronaldo', 'born', 'february', '5', ',', '1985', ',',
'funchal', ',', 'madeira', ',', 'portugal', '.']
```

Phonetic Analysis

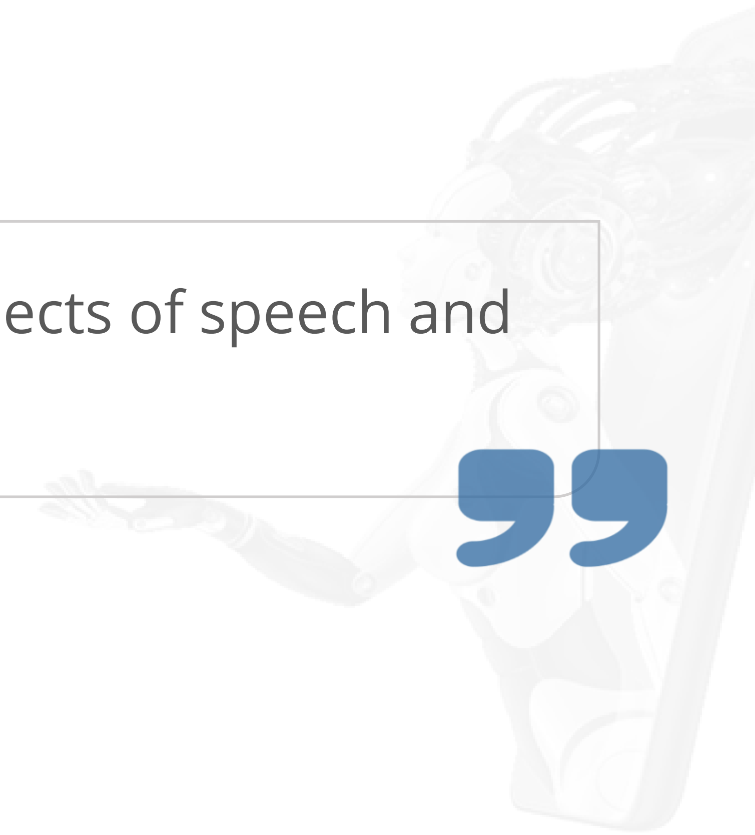
Phonetic Analysis

> Phonetic analysis uses a set of letter-to-sound rules that translate text to phonemes producing usably accurate pronunciations of words and sounds.

Prosodic Analysis

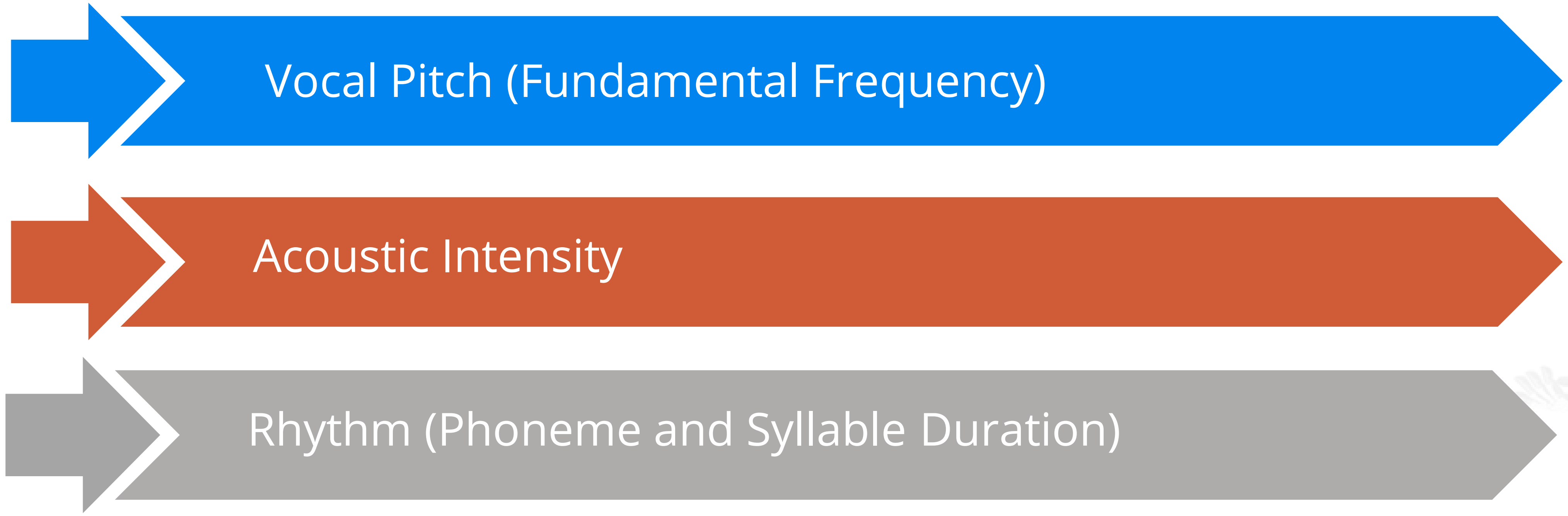
What Is Prosody?

> Prosody is the study of tune and rhythm of speech to understand different aspects of speech and how these features contribute to the meaning.



Characteristics of Prosody

At the phonetic level, prosody is characterized by:



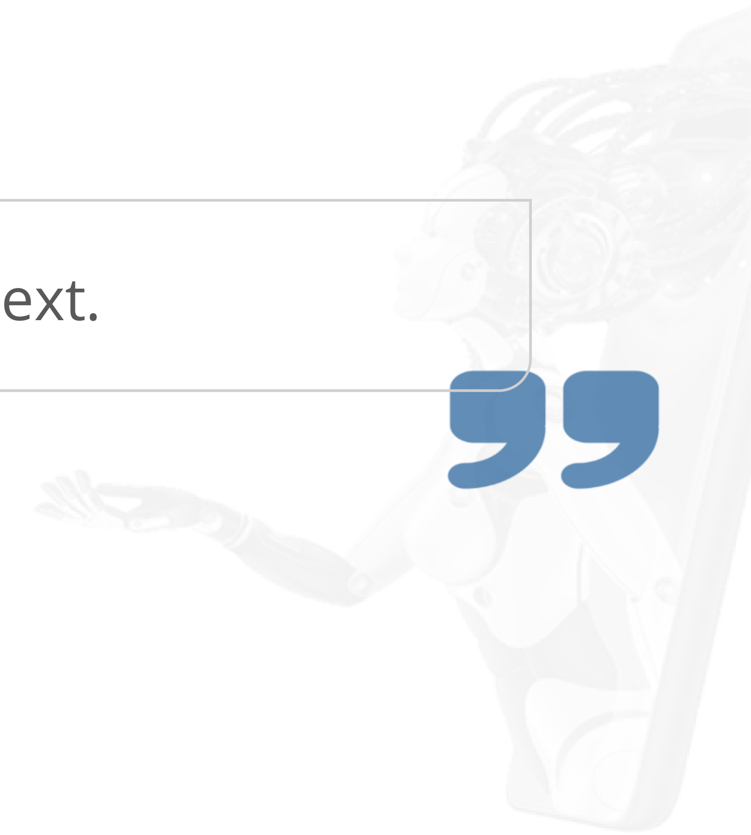
Vocal Pitch (Fundamental Frequency)

Acoustic Intensity

Rhythm (Phoneme and Syllable Duration)

Prosodic Analysis

> It is the analysis performed for TTS on the basis of prosody of the text.



Prosodic Analysis

- Import **prosodic** and create a text object

Code

```
import prosodic as p

text = p.Text(string_or_filename)
```

Prosodic Analysis

- Parse the text object metrically and save the stats

Code

```
text.parse()

text.save_stats()
```

Prosodic Analysis

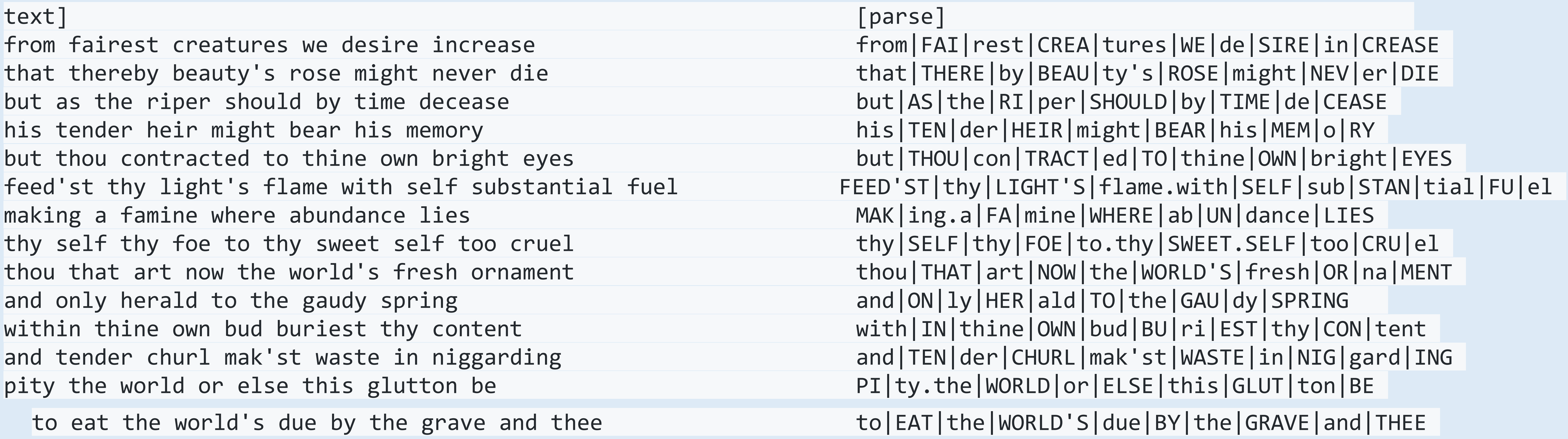
- Iterate over the features

Code

```
for line in text.lines():
    best_parse = line.bestParse()   # most plausible parse
    all_parses = line.allParses()   # all plausible parse
    first_word = line.words()[0]
    last_syllable = line.syllables()[-1]
    last_syllable_rime = line.rimes()[-1]
    last_syllable_rime_phonemes = last_syllable_rime.phonemes()
```

Prosodic Analysis

Output



```
text]                                              [parse]
from fairest creatures we desire increase          from|FAI|rest|CREA|tures|WE|de|SIRE|in|CREASE
that thereby beauty's rose might never die         that|THERE|by|BEAU|ty's|ROSE|might|NEV|er|DIE
but as the riper should by time decease            but|AS|the|RI|per|SHOULD|by|TIME|de|CEASE
his tender heir might bear his memory              his|TEN|der|HEIR|might|BEAR|his|MEM|o|RY
but thou contracted to thine own bright eyes       but|THOU|con|TRACT|ed|TO|thine|OWN|bright|EYES
feed'st thy light's flame with self substantial fuel  FEED'ST|thy|LIGHT'S|flame.with|SELF|sub|STAN|tial|FU|el
making a famine where abundance lies               MAK|ing.a|FA|mine|WHERE|ab|UN|dance|LIES
thy self thy foe to thy sweet self too cruel       thy|SELF|thy|FOE|to.thy|SWEET.SELF|too|CRU|el
thou that art now the world's fresh ornament       thou|THAT|art|NOW|the|WORLD'S|fresh|OR|na|MENT
and only herald to the gaudy spring                and|ON|ly|HER|ald|TO|the|GAU|dy|SPRING
within thine own bud buriest thy content           with|IN|thine|OWN|bud|BU|ri|EST|thy|CON|tent
and tender churl mak'st waste in niggarding        and|TEN|der|CHURL|mak'st|WASTE|in|NIG|gard|ING
pity the world or else this glutton be             PI|ty.the|WORLD|or|ELSE|this|GLUT|ton|BE
  to eat the world's due by the grave and thee     to|EAT|the|WORLD'S|due|BY|the|GRAVE|and|THEE
```

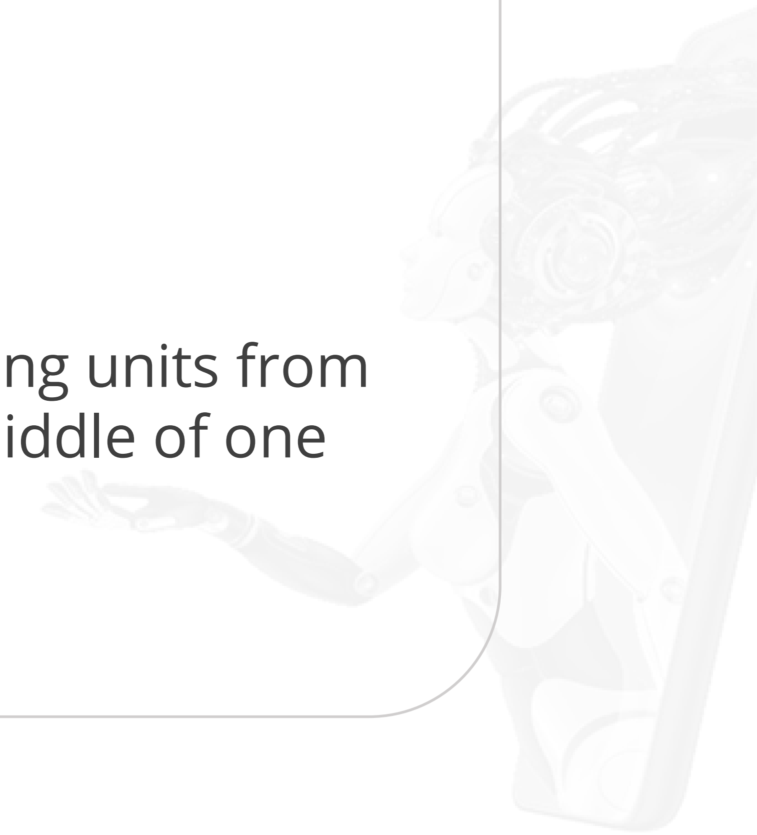
Waveform Synthesis

Waveform Synthesis

> It is a sound synthesis technique used to create periodic waveforms.

Diphone Concatenative Synthesis

- Is a sub process of waveform synthesis
- Generates a waveform from a sequence of phones by selecting and concatenating units from a prerecorded database of diphone (a phone-like unit going from roughly the middle of one phone to the middle of the other phone)

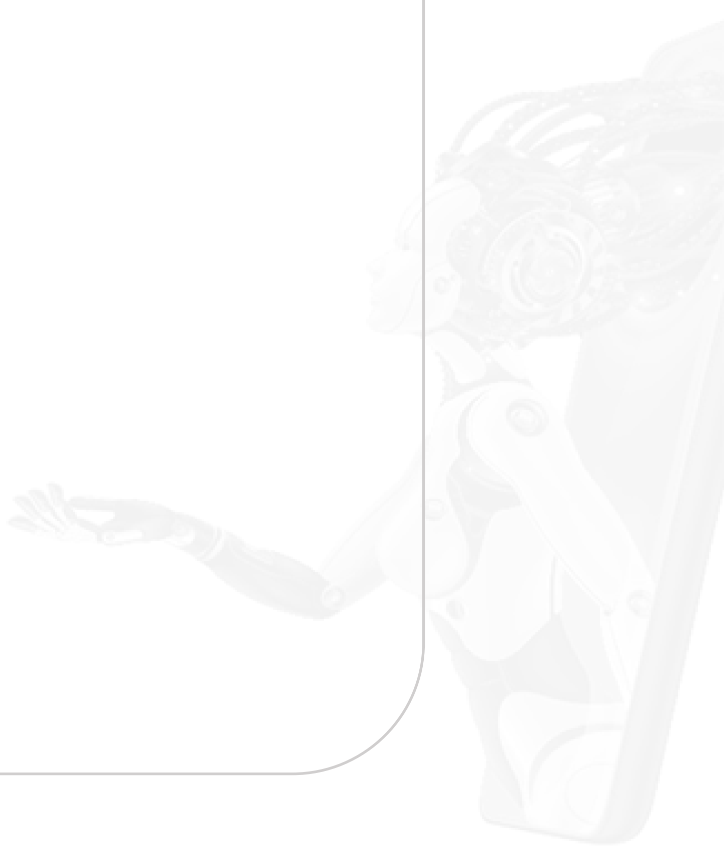


Diphone Concatenative Synthesis

It is characterized by the following steps:

Training:

- Record a single speaker with a verbal example of each diphone
- Cut out each diphone from the speech and store them in a diphone database



Diphone Concatenative Synthesis

Synthesis:

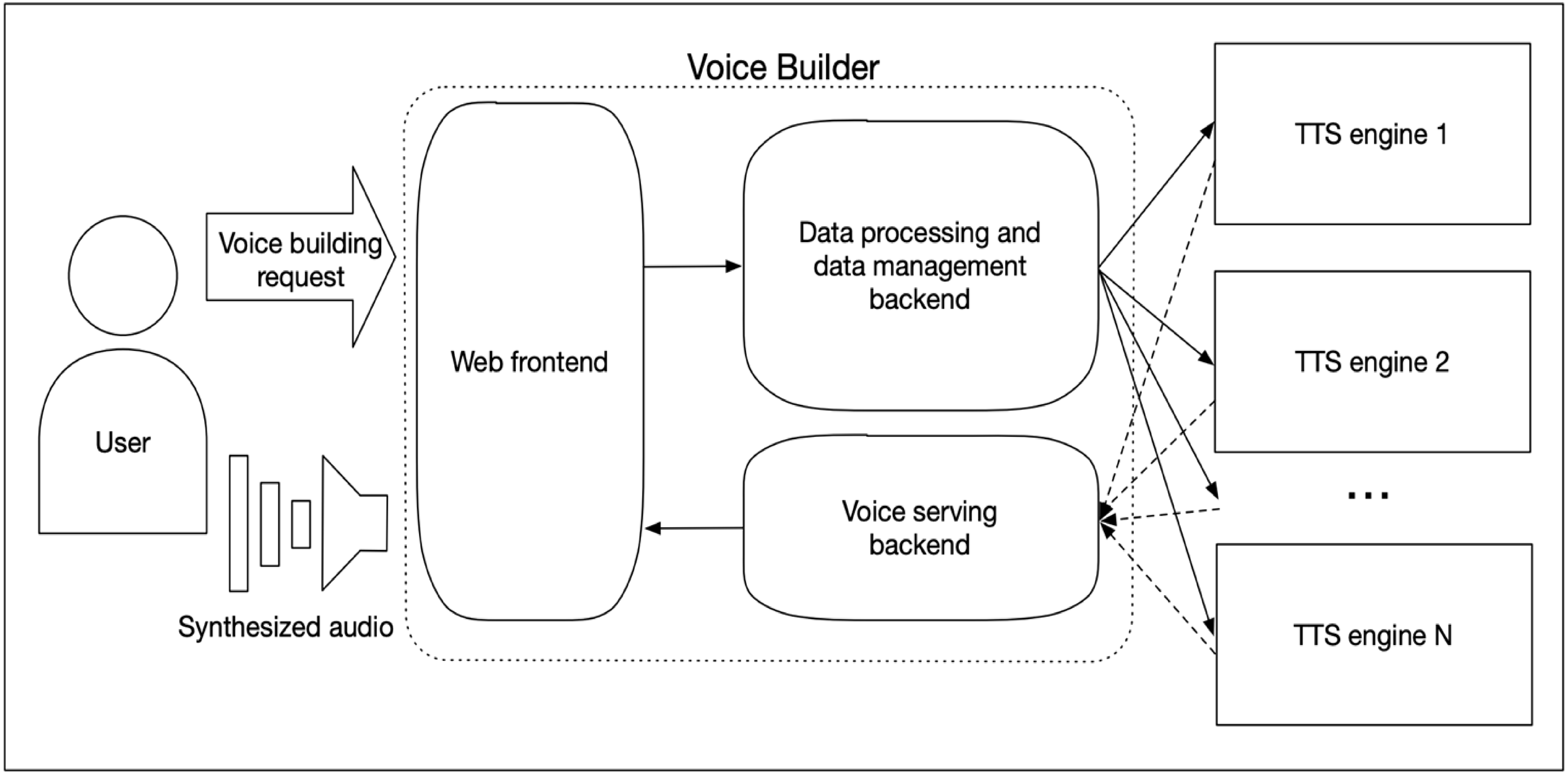
- Take a sequence of diphones that corresponds to the desired phone sequence from the database
- Concatenate the diphones through slight signal processing at the boundaries
- Use signal processing to change the prosody (f 0, duration) of the diphone sequence to desired prosody

Voice Builder

Voice Builder

> It is a TTS voice-building tool designed to allow users quickly build and listen to initial voices. It consists of a web frontend that allows users to synthesize voices, regardless of their technical ability.

Voice Builder



Modules for Text to Speech Conversion

Modules for Text to Speech Conversion

- pyttsx3
- gTTS (Google Text to Speech)

pyttsx3

> It is a python module that synthesizes text to speech. This package works in Windows, Mac, and Linux. It uses available native speech drivers and works completely offline.

Installation of pyttsx3

- **pyttsx3** is installed with pip

Code

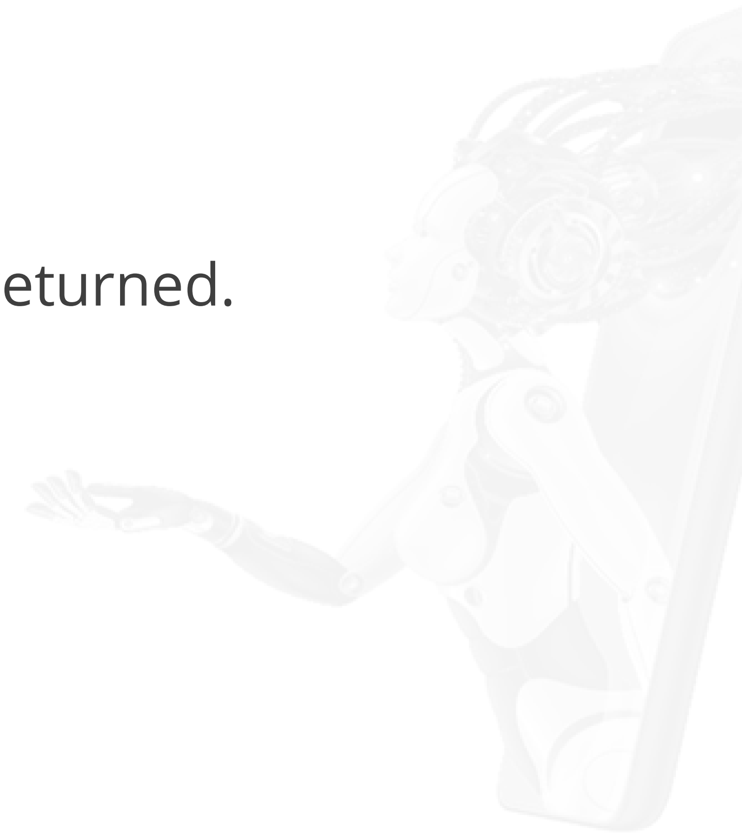
```
pip install pyttsx3
```

- It is recommended to install **win32** to avoid unexpected error because **pyttsx3** is dependent on **win32** library.

```
pip install win32
```

Important Functions of pyttsx3

- **pyttsx3.init([driverName : string, debug : bool])**
  - Gets a reference to an engine instance that will use the given driver
  - If the requested driver is already in use by another engine instance, that engine is returned. Otherwise, a new engine is created
- **getProperty(name : string)**
  - Gets the current value of an engine property



Important Functions of pyttsx3

- **setProperty(name, value)**
    - Queues a command to set an engine property
    - The new property value affects all utterances queued after this command

- **say(text : unicode, name : string)**
    - Queues a command to speak an utterance
    - The speech is the output according to the properties set before this command in the queue

Important Functions of pyttsx3

runAndWait()

- Blocks while processing all currently queued commands
- Invokes callbacks for engine notifications appropriately
- Returns engine instance when all commands queued before the call are emptied from the queue

Use of pyttsx3

- Speaking text:

Code

```
import pyttsx3
engine = pyttsx3.init()
engine.say('I am on seafood diet. I see food and eat it.')
engine.runAndWait()
```

Use of pyttsx3

- Listening for events:

Code

```
import pyttsx3
def onStart(name):
   print 'starting', name
def onWord(name, location, length):
   print 'word', name, location, length
def onEnd(name, completed):
   print 'finishing', name, completed
engine = pyttsx3.init()
engine.connect('started-utterance', onStart)
engine.connect('started-word', onWord)
engine.connect('finished-utterance', onEnd)
engine.say(‘I am on seafood diet. I see food and eat it.')
engine.runAndWait()
```

Use of pyttsx3

- Interrupting an utterance:

Code

```
import pyttsx3
def onWord(name, location, length):
   print 'word', name, location, length
   if location > 10:
      engine.stop()
engine = pyttsx3.init()
engine.connect('started-word', onWord)
engine.say('The quick brown fox jumped over the lazy dog.')
engine.runAndWait()
```

Use of pyttsx3

- Changing voices:

Code

```
engine = pyttsx3.init()
voices = engine.getProperty('voices')
for voice in voices:
   engine.setProperty('voice', voice.id)
   engine.say('The quick brown fox jumped over the lazy dog.')
engine.runAndWait()
```

Use of pyttsx3

- Changing speech rate:

Code

```
engine = pyttsx3.init()
rate = engine.getProperty('rate')
engine.setProperty('rate', rate+50)
engine.say(‘I am in seafood diet. I see food and eat it.')
engine.runAndWait()
```

Use of pyttsx3

- Changing volume:

Code

```
engine = pyttsx3.init()
volume = engine.getProperty('volume')
engine.setProperty('volume', volume-0.25)
engine.say('The quick brown fox jumped over the lazy dog.')
engine.runAndWait()
```

Use of pyttsx3

- Running a driver event loop:

Code

```
engine = pyttsx3.init()
def onStart(name):
   print 'starting', name
def onWord(name, location, length):
   print 'word', name, location, length
def onEnd(name, completed):
   print 'finishing', name, completed
   if name == 'fox':
      engine.say('What a lazy dog!', 'dog')
   elif name == 'dog':
      engine.endLoop()
engine = pyttsx3.init()
engine.connect('started-utterance', onStart)
engine.connect('started-word', onWord)
engine.connect('finished-utterance', onEnd)
engine.say('The quick brown fox jumped over the lazy dog.', 'fox')
engine.startLoop()
```

Use of pyttsx3

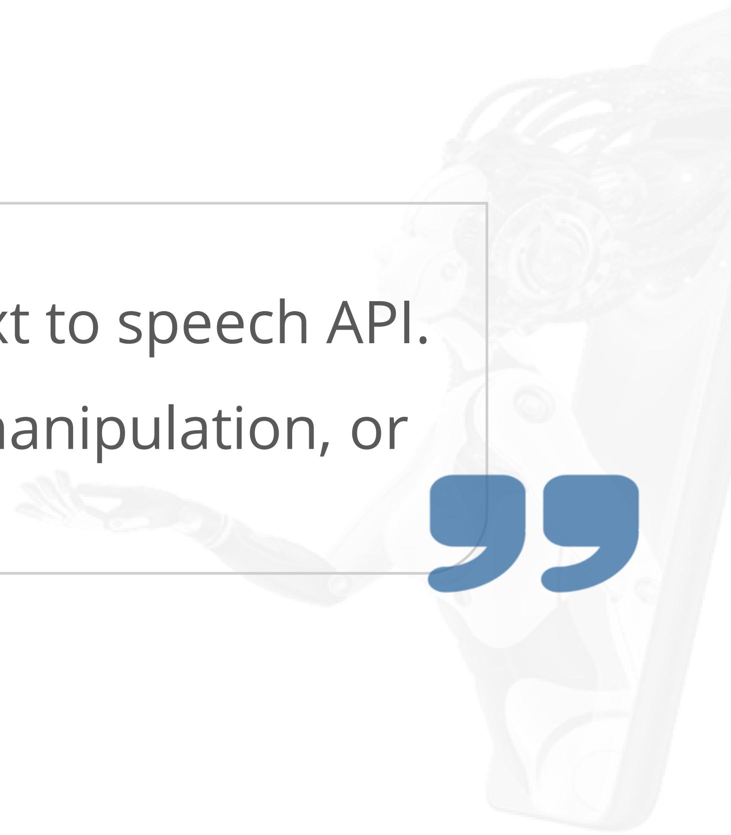
- Using an external event loop:

Code

```
engine = pyttsx3.init()
engine.say('The quick brown fox jumped over the lazy dog.', 'fox')
engine.startLoop(False)
# engine.iterate() must be called inside externalLoop()
externalLoop()
engine.endLoop()
```

gTTS

> It is a Python library and command-line tool to interface with Google Translate's text to speech API. It writes spoken mp3 data to a file, a file-like object (bytestring) for further audio manipulation, or **stdout**.



Features of gTTS

Customizes speech-specific sentence tokenizer that allows unlimited lengths of text to be read, all the while keeping proper intonation, abbreviations, and decimals

Customizes text preprocessors which can provide pronunciation corrections

Retrieves supported languages automatically

Installation of gTTS

- **gTTS** is installed with pip:

Code

```
pip install gTTS
```

gTTS Command-Line

- After installing the package, the **gtts-cli** tool becomes available. With the following code its existence can be checked:

Code

```
gtts-cli
```

Example of gTTS Command-Line

- List available languages:

Code

```
gtts-cli --all
```

Example of gTTS Command-Line

- Read “hello” to hello.mp3:

Code

```
gtts-cli 'hello' --output hello.mp3
```

Example of gTTS Command-Line

- Read “bonjour” in French to bonjour.mp3:

Code

```
gtts-cli 'bonjour' --lang fr --output bonjour.mp3
```

Example of gTTS Command-Line

- Read “c’est la vie” in French to **cestlavie.mp3**:

Code

```
gtts-cli "c'est la vie" --lang fr --output cestlavie.mp3
```

Example of gTTS Command-Line

- Read "slow" slowly to **slow.mp3**:

Code

```
$ gtts-cli "slow" --slow --output slow.mp3
```

Example of gTTS Command-Line

- Read "hello" to **stdout**:

Code

```
gtts-cli "hello"
```

Example of gTTS Command-Line

- Read **stdin** to **hello.mp3** via <text> or <file>:

Code

```
echo -n "hello" | gtts-cli - --output hello.mp3

echo -n "hello" | gtts-cli --file - --output hello.mp3
```

Example of gTTS Command-Line

- Read "no check" to **nocheck.mp3** without language checking:

Code

```
gtts-cli "no check" --lang zh --nocheck --ouput nocheck.mp3
```

Example of gTTS Command-Line

- Play the song directly:

Code

```
gtts-cli "hello" | play -t mp3 -
```

Example of gTTS Module

- Write "hello" in English to **hello.mp3**:

Code

```
from gtts import gTTS
tts = gTTS("hello", lang="en")
tts.save("hello.mp3")
```

Example of gTTS Module

- Write "hello bonjour" in English then into French as **hello_bonjour.mp3**:

Code

```
from gtts import gTTS
tts_en = gTTS("hello", lang="en")
tts_fr = gTTS("bonjour", lang="fr")

with open("hello_bonjour.mp3", "wb") as f:
      tts_en.write_to_fp(f)
      tts_fr.write_to_fp(p)
```

Example of gTTS Module

- Instead of writing to disk, get the URL for "hello" in English:

Code

```
from gtts import gTTS
tts = gTTS("hello", lang="en")
tts.get_urls()
```

Logging of gTTS Module

- **gTTs** does logging using the standard Python logging module. The following loggers are used:

gtts.tts	Used for the **gTTS** class
gtts.lang	Used for the **lang** module (language fetching)
gtts	Is an upstream logger for all of the above

Convert Text to Speech Using gTTS

Problem Statement : Audiobook industry is booming nowadays. It’s not feasible to record each and every book by a voice-over artist. You have to come up with a simple program to automate the conversion of text to speech.

Objective: Convert text file and a line of text to speech using **gTTS** python module.

Access: Click the Practice Labs tab on the left panel. Now, click on the START LAB button and wait while the lab prepares itself. Then, click on the LAUNCH LAB button. A full-fledged Jupyter lab opens, which you can use for your hands-on practice and projects.

ASSISTED PRACTICE

Convert Text to Speech Using pyttsx

Problem Statement : You work in a content marketing business and convert the script to voice for video content using a cloud-based converter. Due to unavoidable reasons, the internet is not working. There is lot of demand from the clients for the delivery to be made in time. You have to use a python library that works offline to convert text to speech.

Objective: Convert text file and a line of text to speech using **pyttsx** python module.

Access: Click the Practice Labs tab on the left panel. Now, click on the START LAB button and wait while the lab prepares itself. Then, click on the LAUNCH LAB button. A full-fledged Jupyter lab opens, which you can use for your hands-on practice and projects.

ASSISTED PRACTICE

Knowledge Check

Knowledge Check

1

Which of the following analyses is a part of text to speech synthesizer architecture?

a. Text Analysis

b. Phonetic Analysis

c. Prosodic Analysis

d. All the above

Knowledge Check

1

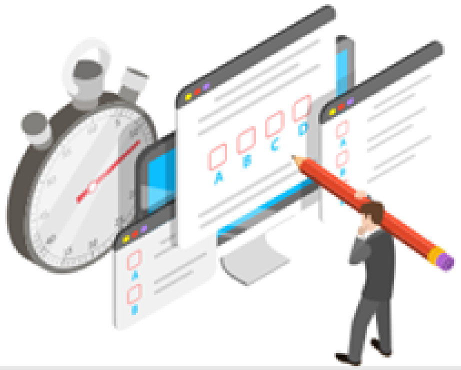
Which of the following analyses is a part of text to speech synthesizer architecture?

a. Text Analysis

b. Phonetic Analysis

c. Prosodic Analysis

d. All the above



The correct answer is **d**

Text analysis, phonetic analysis, and prosodic analysis along with speech synthesis are part of text to speech synthesizer architecture.

Knowledge Check

2

At the phonetic level, ______ is characterized by vocal pitch.

a. Prosody Analysis

b. Text Analysis

c. Both a and b

d. None of the above

Knowledge Check

2

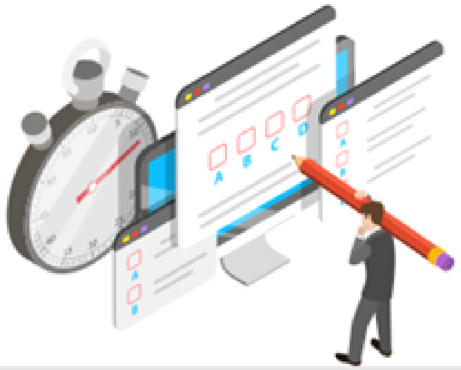
At the phonetic level, ______ is characterized by vocal pitch.

a. Prosody Analysis

b. Text Analysis

c. Both a and b

d. None of the above



The correct answer is **a**

At the phonetic level, prosody analysis is characterized by vocal pitch.

Knowledge Check

3

Which of the following processes breaks strings into small structures or units?

a. Tokenization

b. Lemmatization

c. Both a and b

d. None of the above

Knowledge Check

3

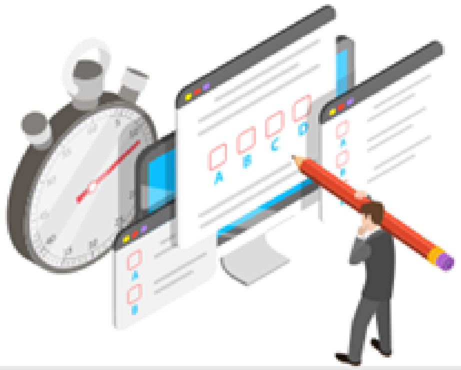
Which of the following processes breaks strings into small structures or units?

a. Tokenization

b. Lemmatization

c. Both a and b

d. None of the above



The correct answer is **a**

Tokenization breaks strings into small structures or units.

Knowledge Check

4

__________ usually refers to normalizing words into its base form or root form.

a. Tokenization

b. Lemmatization

c. Stemming

d. None of the above

Knowledge Check

4

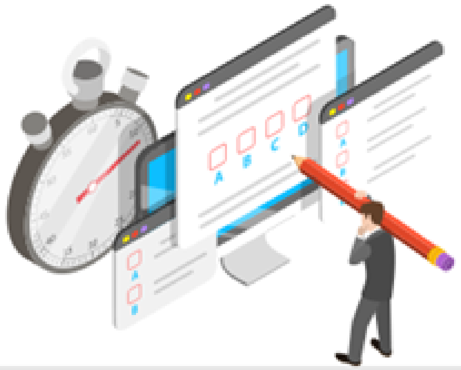
__________ usually refers to normalizing words into its base form or root form.

a. Tokenization

b. Lemmatization

c. Stemming

d. None of the above



The correct answer is **c**

Stemming usually refers to normalizing words into its base form or root form.

Knowledge Check

5

Which of the following is the python module to convert text to speech?

a. gtts

b. pyttsx

c. Both a and b

d. None of the above

Knowledge Check

5

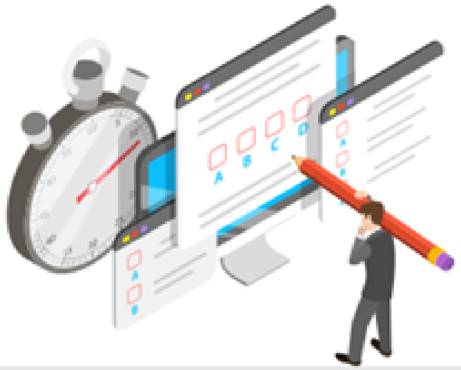
Which of the following is the python module to convert text to speech?

a. gtts

b. pyttsx

c. Both a and b

d. None of the above



The correct answer is **c**

Both “gtts” and “pyttsx” are widely used python modules to convert text to speech.

Knowledge Check

6

Which of the following libraries is imported to perform prosodic analysis on a statement?

a. prosodic

b. prosody

c. pros

d. None of the above

Knowledge Check

6

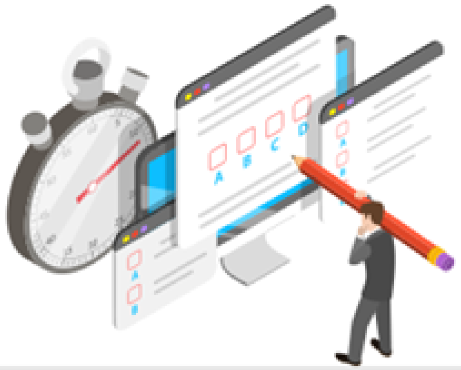
Which of the following libraries is imported to perform prosodic analysis on a statement?

a. prosodic

b. prosody

c. pros

d. None of the above



The correct answer is **a**

By importing "prosodic" library as "import prosodic", prosodic analysis is performed on a statement.

Make a Graphical User Interface to Convert Text to Speech

Problem Scenario:

Joe works for a famous instagram influencer, who makes lot of podcasts. Instead of voicing them over personally, he wants Joe to make a graphical user interface, where text can be typed and the script can be received as an output.

Objective:

Create a graphical user interface to convert text to speech using **pyttsx** and **Tkinter** in Python.

Access:

Click the Practice Labs tab on the left panel. Now, click on the START LAB button and wait while the lab prepares itself. Then, click on the LAUNCH LAB button. A full-fledged Jupyter lab opens, which you can use for your hands-on practice and projects.

LESSON-END PROJECT

Key Takeaways

- Text analysis is a process that allows machines to extract and classify information from text like tweets, emails, support tickets, product reviews, survey responses, etc.

- Waveform synthesis is a technique used to create periodic waveforms.