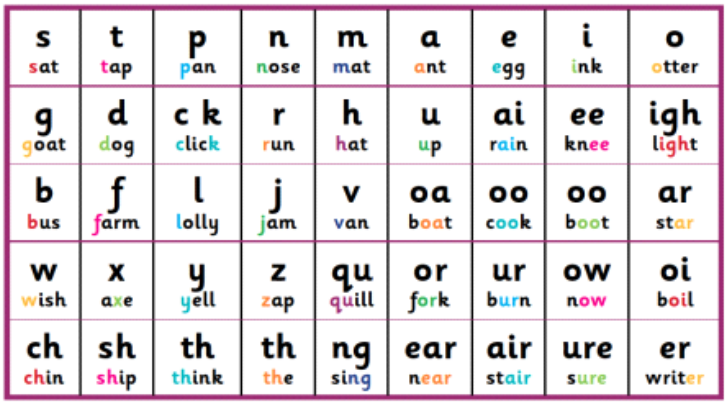
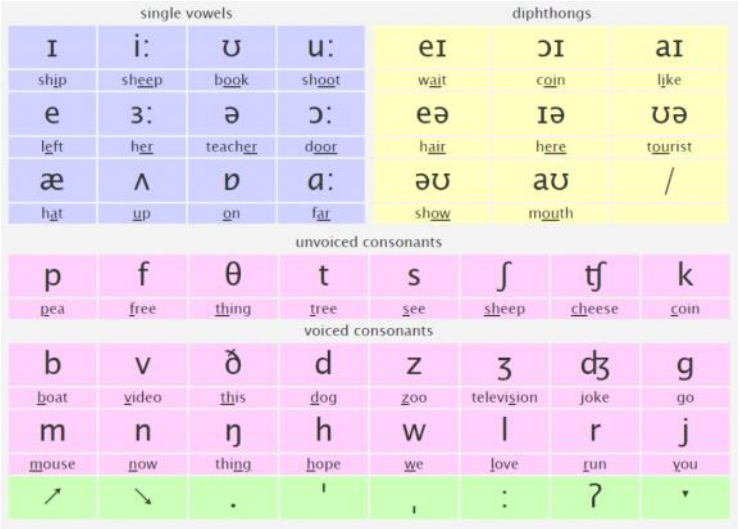
TTS Session 1

٢٣ يناير، ٢٠٢٢ ٠٩:٤٩ م

- **Session 1**
  - **Resources**
    - Lectures by Professor Kishore Prahallad
      - Classification of Speech Sounds lecture
      - Speech features in time domain
      - Speech features in frequency domain
      - Cepstrum and Mel-Frequency
      - Text-to-speech (TTS) - A Quick Overview lecture
    - Festival manual
  - **Terminology**
    - **Phoneme**: The smallest unit of sound, if we change a phoneme in the pronunciation of a word we get a different word
      - A word can be written in broad transcription system by the IPA (International Phonetic Association), we use slashes / / to specify the phoneme of a word
        - the word puff in broad transcription is /pʌf/
      - there are 44 phonemes in English
        - https://www.theschoolrun.com/what-is-a-phoneme



        - https://www.preptoz.com/library/learn-english-pronunciation-phonemic-chart/

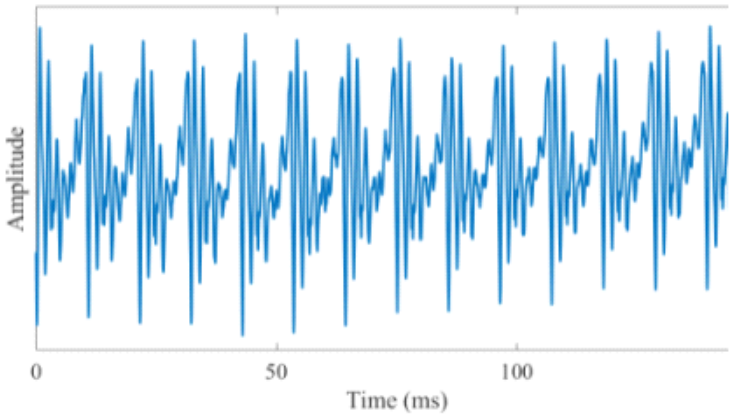


        - Each language has its own basic set of phonemes. For example English has 44 phonemes and Hindi has 47 phonemes
        - A phoneme is different from letter

- Some languages has one-to-one correspondance between a letter and a phoneme such as indian, while for English there is no direct relation between a letter and a phone
  - For example in Englsih the letter c is pronuonced as /k/ and /ch/ in "catch" and the letter k is not pronuouncd in "knight", so we need letter to sound rules for these languages
- Diphone: adjacent pair of phones,
  - For example, in the word "diphon" the phones are: d, a, ɪ, f, ə, ʊ, n. While the diphones are [da], [aɪ], [ɪf], [fə], [əʊ], [ʊn].

- **Prosody**: Prosody, or the way things are spoken, is an extremely important part of the speech message. Changing the placement of emphasis in a sentence can change the meaning of a word, and this emphasis might be revealed as a change in pitch, volume, voice quality, or timing. Prosody is very similar to intonation and is sometimes used interchangabely with it.
  - Prosody consists of the following
    - Pauses: for example pausing between phrases in the sentence (i.e: "He said that [pause] he would go to the school"
      - Prosodic phrasing: is the means by which speakers of any given language break up an utterance into meaningful chunks.
        - prosodic phrases literally signal the end of one tune and the beginning of another
      - Phrases:
        - Definition: a word or group of words forming a syntactic constituent with a single grammatical function
          - Phrases are groups of words that act as a part of speech but cannot stand alone as a sentence, For example, phrases can function as nouns, verbs, adjectives, or adverbs.
        - Examples
          - Noun phrases: a cute baby, many of the theories
          - Verb phrases: has been eating
          - Adjective phrases: not very healthy, terribly long
          - Adverb phrases: slowly and surely, beautifully
          - Prepositional phrases: after a long time, on the table
    - Fundamental frequency (pitch contours): fundamental frequency changes over speech, also referred to as **intonation**
      - Fundamental frequency (F0)
        - Aalto university
          - Fundamental frequency (F0) refers to the approximate frequency of the (quasi-) periodic structure of voiced speech signals, it

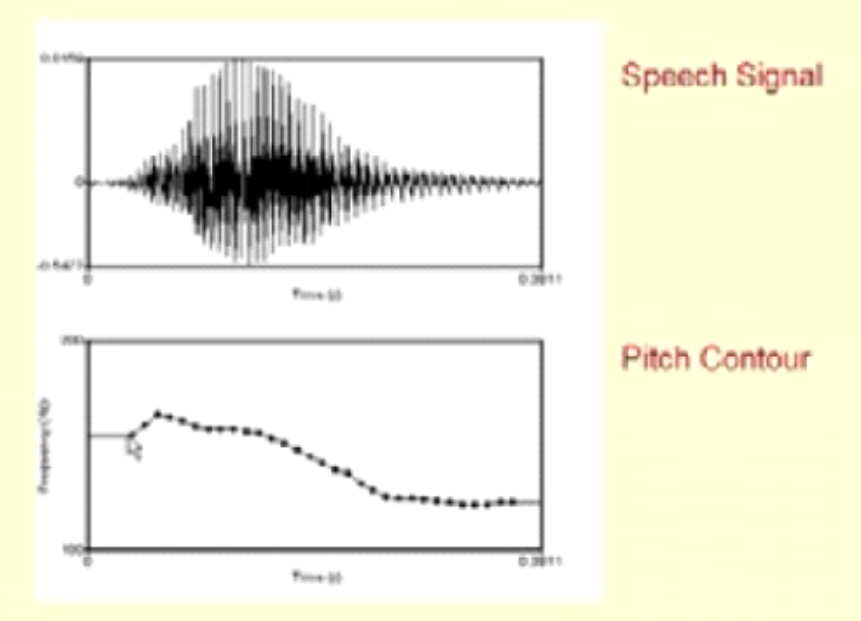
is defined as the average number of oscillations per second and expressed in Hz



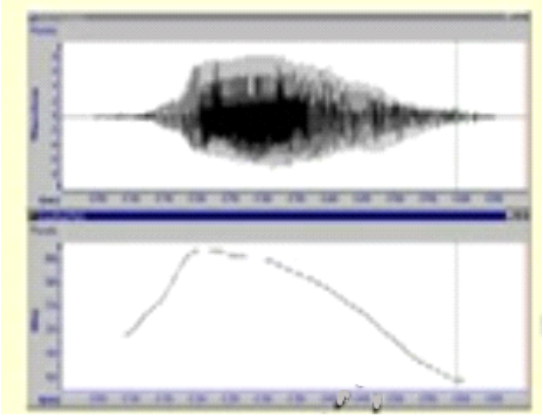
A speech signal with a fundamental frequency of approximately F0=93Hz.

- The oscillation originates from the vocal folds, it isn't perfectly periodic , it has amount of variation in period (known as jitter) and amount of variation in amplitude (known as shimmer)
  - Moreover, the F0 is typically not stationary, but changes constantly within a sentence. In fact, the F0 can be used for expressive purposes to signify, for example, emphasis and questions.
  - Typically fundamental frequencies lie roughly in the range 80 to 450 Hz, where males have lower F0 than females and children [Male: 50 - 200 Hz, Female 200 - 450 Hz]. The F0 of an individual speaker depends primarily on the length of the vocal folds, which is in turn correlated with overall body size.
- The fundamental frequency is closely related to pitch, which is defined as our perception of fundamental frequency. That is, the F0 describes the actual physical phenomenon, whereas pitch describes how our ears and brains interpret the signal,

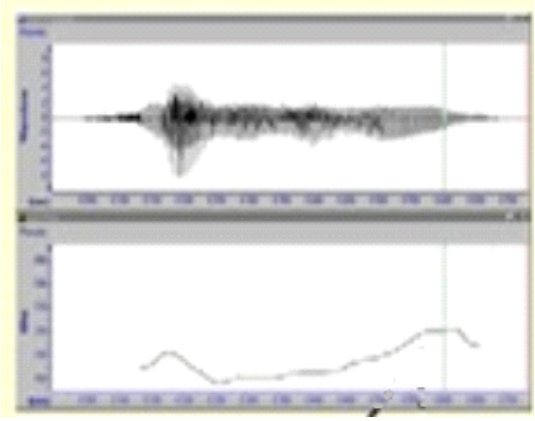
- Fundamental frequency is observed only in voiced regions of sound
  - for unvoiced regions we usually assign a random variable for pitch
- Pitch contour (sometimes referred to by intonation)
  - Plot of fundamental frequencies over smal analysis windows (remember that fundamental frequency isn't stationary but changes during the speech)



- The change in pitch contour may reflect style of saying sentences:
  - For passive sentences (i.e *She is lying*), the pitch contour is falling



  - For questions (i.e *Is she lying?*) the pitch contour is rising

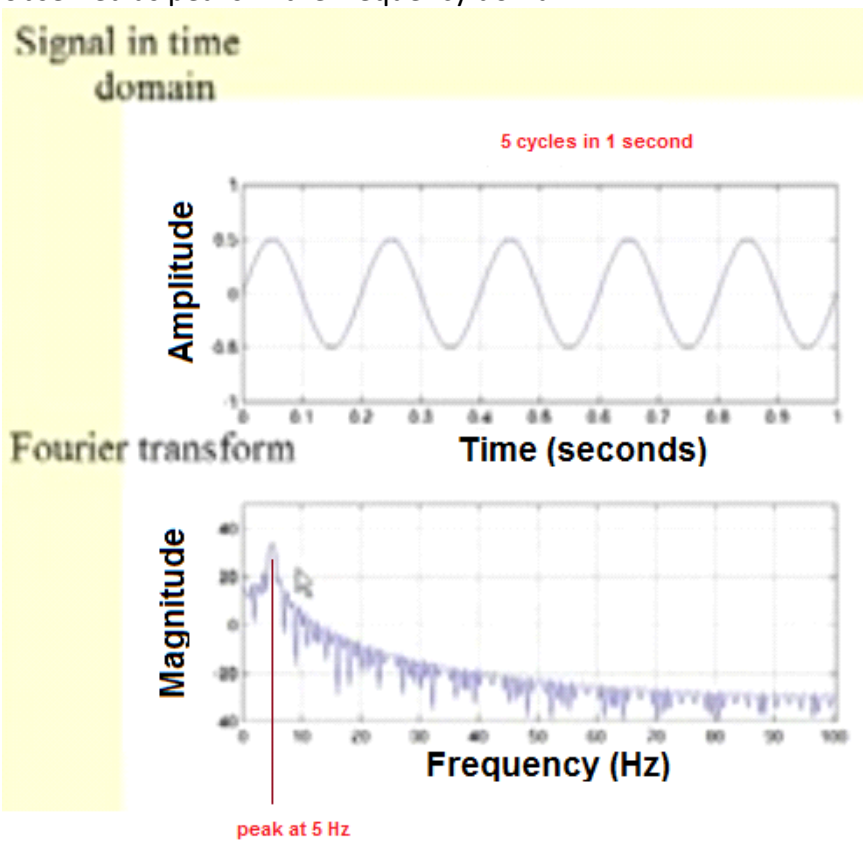


- Pitch plays important role in the naturalness of he synthesized speech, also it is useful for gender identification
- Duration of each phone
- loudness

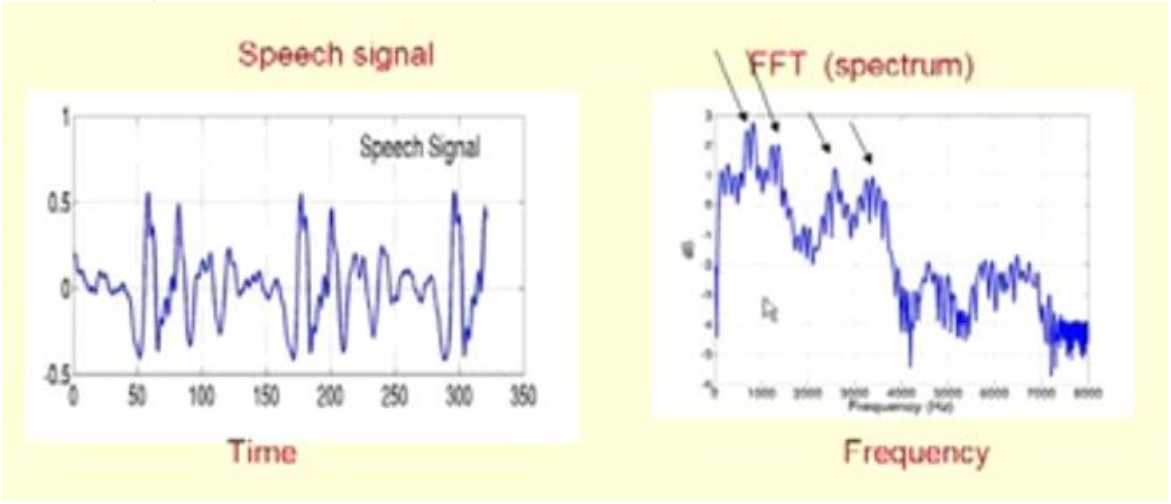
- **Formants:**
  - Fourier transform
    - Takes the signal in time domain and gives you the frequency

components of the signal (called the spectrum of the signal or the signal in frequency domain)

- The dominant frequency components in the signal are observed as peaks in the frequency domaim



- Speech and its spectrum



- The peaks in speech spectrum are called **formants**
  - Each sound in a word has its own distictive formants
    - The /ih/ in bit has formants at 400 Hz, 1900 Hz and 2550 Hz
    - This can help in speech recognition: seeing peaks at specific frequencies can tell us the sound pronounded in the word

- **Part of speech**: For example in the English language: noun, pronoun, verb, adjective, adverb, preposition, ...
  - part of speech tagging is the process of assigning a POS tag to each word in a sentence.

- **Text to Speech**
  - Utterance: a term that represent some chunk of text that is to be

rendered as speech. In general you may think of it as a sentence that we want to synthesize as speech.
    - Separating a text into utterances is important, as it allows synthesis to work bit by bit, allowing the waveform of the first utterance to be available more quickly than if the whole files was processed as one.
- Tokenization: Converting the string of characters into a list of tokens (typically by white space splitting)
- Text normalization (Token to word): Convert each tokens to zero or more words, expanding numbers, abbreviations, measure units, dates, symbols (such as dollar symbols), country abbreviations
    - 12 -> "twelve"
    - Dr. Ben -> Doctor Ben
    - $ 14 -> "fourteen dollars"
    - 12 in. -> "twelve inches"
- Linguistic Feature extraction:
    - Prosodic phrasing: Chunk utterance into prosodic phrases.
    - Part of speech assigning a POS tag to each word in a sentence.
    - Intonation: decide the fundamental frequency contour of the tokens, for example the end part
- Lexical lookup: given a word, output sequence of phonemes (find its pronounciation)
    - You can use a pronunciation dictionary (lexicon) that maps a word to its phones
    - You can also use a set of rules (called letter to sound rules)
        - For example in indian language when the phone /a/ occurs at the end of a word then delete it (i.e map it to the null phone ($))
        - Leter to sound rules can be learnt using statistical models Classification And Regression Tree CART, Hidden Markov model (HMM), Neural networks
- **Speech Synthesis**: Here we have 2 main approaches
    - Concatenative Approach
        - First thinking
            - Steps
                - record set of phones
                - Given a text, for each word obtain the sequence of phones to be concatenated
                - Concatenate the pre-recorded phones to get speech
            - Output
                - You will get terrible garbage speech
        - Refinement: What do we need to include in our model?
            - Coarticulation (coupling effect): Production of /k/ and /a/ phonemes is different from the sound that should be produced by /ka/
            - Prosody (energy, pitch, duration)
                - Energy contour (amplitude)
                - Pitch and its contour [F0 contour] (variant between phones)
                - Duration (how long each phone should be)

- Solution
  - Record a speech database and select the required unit (a unit can be phone or multiple phones)
    - Choice of a unit
      - Word as a unit: means the database will have the speech waveforms of all words (large database needed and it is difficult to ensure coverage of all possible words).
        - useful for limited domains (such as talking clock, reservation system, ...)
      - Phone as a unit: but there will be the problem of absence of coarticulation
      - Diphone as unit (**Diphone synthesis**): preserves the transition region between phones and thus coarticulation is taken into account
        - widely used, produce clear sound (but not natural)
        - Diphone starts at the middle of the first phone and ends at the middle of the second phone (so as to preserve the transition between phones)
        - How to build diphone database:
          - record all possible phone-phone combinations in a language (exclude combinations that are not allowed (do not occur) in a language)
          - From each of the phone-phone recordings manually label diphone boundaries (may use tools such as Emulabel)
- **Unit selection synthesis**
  - Diphone synthesis database contains single record of the diphone for each diphone in the language, this gives un-natural voice as the same diphone can sound different if it is put in different linguistic context (a diphone at the start of the word will sound slightly diferent than the same diphone but at the middle of a word)
  - In unit selection synthesis, the speech database will contain multiple examples of each diphone in different contexts (and thus we store diphones with varying prosody) and in the synthesis process **select** the diphone with suitable prosody
  - How to build unit selection database
    - Choose 2000 sentences which have a good coverage of all possible diphones. Speak the sentences one by one to create 1-2 hours of speech.
    - Automatic extraction
      - Label the phone boundaries in each of spoken

sentences. this task is called **speech segementation** and can be done using Hidden Markov Models (also neural networks)

- Given the phone boundaries, approximate the diphone boundaries so we now have a set of diphone-like units (diphone-likes because they are approximated automatically not as the before manually precise diphones)

– Indexing:

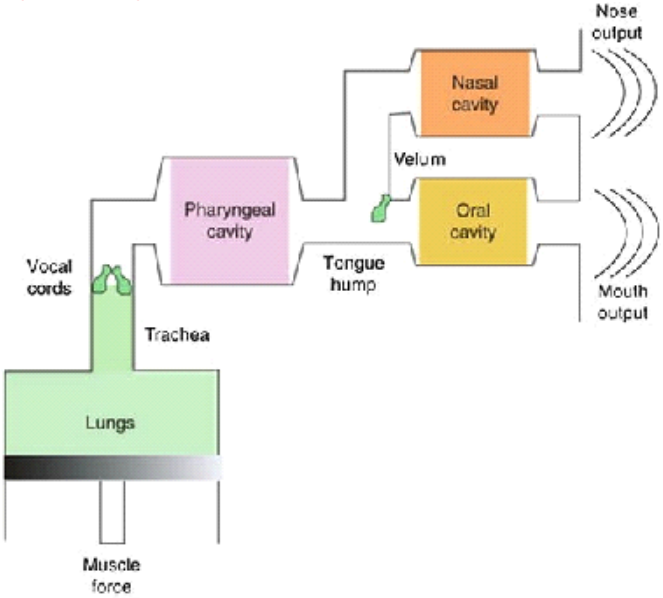
- For each diphone-like unit, store context information: previous phone, next phone, position in the syllable, position in the word, ...
- You will end up with thousands of instances for single diphone-like unit, each unit may differ in the context information
- For each unit build a decision tree to split the instances of the units (thousands) into several sub-clusters

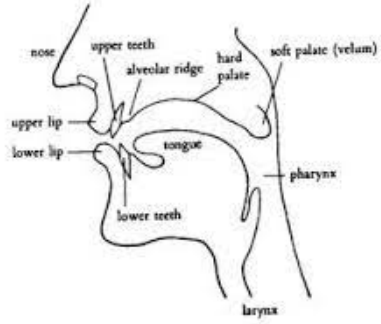
– During synthesis:

- for each diphone traverse through the decision tree to reach the sub-cluster best matching the context of this phone

◆ Statistical Approach: In these approaches we try to mimic the way the sound is actually produced (source and filter model)

◇ How speech is prooduced?

□ 

□ 

– [img src]

□ Speech is produced when the air is expelled from the

lungs through the trachea. The vocal chords are caused to vibrate by the air flow

- The speech production is modeled as
  - a source represented by the vocal folds (vocal cords), the vocal cords operate in 2 modes:
    - open mode: release the air completely
    - vibrating mode: chop the air flow into periodic pulses
  - vocal tract (nasal canvity, oral cavity, pharynx): modulates the air flow by changing the shape of the various cavities (changing mouth opening, tongue position, lips, ...)
- Sounds are produced by varying the shape of the vocal tract and by varying the excitation (source)

- So in statistical approaches we try to train a model to act as the filter during synthesis of a certain word

- **Festival**
  - Festival offers a general framework for building speech synthesis systems, it offers full text to speech through a number APIs: from shell level, though a Scheme command interpreter, as a C++ library, and an Emacs interface
  - The system is written in C++ and uses the Edinburgh Speech Tools Library for low level architecture and has a Scheme (SIOD) based command interpreter for control
  - Installation
    - Requirements
      - festival-1.4.3-release.tar.gz: Festival Speech Synthesis System source
      - speech_tools-1.2.3-release.tar.gz: Edinburgh Speech Tools Library
      - festlex_NAME.tar.gz: The lexicon distribution, where possible, includes the lexicon input file as well as the compiled form (OALD is free for non-commercial use)
      - festvox_NAME.tar.gz: speech database (Each voice may have other dependencies such as requiring particular lexicons)
      - festdoc_1.4.3.tar.gz: documentation for Festival and the Speech Tools
      - C++ compiler: : GCC 2.7.2 plus egcs (from Cygnus GNU win32 b19), Visual C++ PRO v5.0, Visual C++ v6.0
        - Test it with compiling he following program

```
#include <iostream.h>
int main (int argc, char **argv)
{
  cout << "Hello world\n";
}
```

      - GNU make
      - Audio hardware: NCD's NAS (formerly called netaudio which is network transparent audio system) and a method allowing arbitrary UNIX commands.
    - My Notes

- Used Cygwin on Windows 8.1 with the following installed packages
  - gcc
  - make
  - grep
  - awk
  - sed
  - libncurses-devel
    - Hint: You can specify what packages to install during Cygwin setup and you can aslo use apt-cyg to install packages, Follow this to install apt-cyg
- Your System PATH needs to include Cygwin bin folder + festival and speech tools bin folders
- Use the command "make info" in the speech tool directory
- in speech tools\config\config: SYSTEM_TYPE=ix86 _CYGWIN32
- in config change -lcurses to -lncurses
- Use the command "make" in the speech tool directory
- After Compiling the speech tools, go and compile festival

- Quick start
  - Festival works in two fundamental modes, command mode and text-to-speech mode
    - In command mode (default mode), information (in files or through standard input) is treated as commands and is interpreted by a Scheme interpreter
    - In tts-mode, information (in files or through standard input) is treated as text to be rendered as speech (by using --tts option)
- As Festival includes the SIOD Scheme interpreter most standard Scheme commands work
  - (SayText "Hello world")
  - Festival may also synthesize from files rather than simply text.
    - (tts "myfile" nil)
- Phonesets
  - A phoneset is a set of symbols which may be further defined in terms of features, such as vowel/consonant, place of articulation for consonants, type of vowel etc.
  - The lexicons, letter to sound rules, waveform synthesizers, etc. all require the definition of a phoneset before they will operate.
  - A phoneset definition has the form
    - (defPhoneSet NAME FEATUREDEFS PHONEDEFS)
      - NAME is any unique symbol used e.g. mrpa, darpa, etc.
      - FEATUREDEFS is a list of definitions each consisting of a feature name and its possible values. For example
        - (
          (vc + -) ;; vowel consonant
          (vlength short long diphthong schwa 0) ;; vowel length
          ...
)
      - PHONEDEFS is a list of phone definitions themselves,

Each phone definition consists of a phone name and the values for each feature defined earlier in the FEATUREDEFS

- Note the phoneset should also include a definition for any silence phones through the command `PhoneSet.silences`. There may be more than one silence phone.
  - (PhoneSet.silences '(#))
- A typical example of a phoneset definition can be found in 'lib/mrpa_phones.scm'. each Phone is caracterized by 8 features (vc [vowel or constant], vlng [vowel length], ...)
  - (defPhoneSet
    mrpa
    ;;; Phone Features
    (;; vowel or consonant
    (vc + -)
    ;; vowel length: short long dipthong schwa
    (vlng s l d a 0)
    ;; vowel height: high mid low
    (vheight 1 2 3 0)
    ;; vowel frontness: front mid back
    (vfront 1 2 3 0)
    ;; lip rounding
    (vrnd + - 0)
    ;; consonant type: stop fricative affricate nasal lateral approximant
    (ctype s f a n l r 0)
    ;; place of articulation: labial alveolar palatal labio-dental
    ;; dental velar glottal
    (cplace l a p b d v g 0)
    ;; consonant voicing
    (cvox + - 0)
)
    ;; Phone set members
    (
    (uh + s 2 3 - 0 0 0)
    (e + s 2 1 - 0 0 0)
    (a + s 3 1 - 0 0 0)
    ...)
)
    (PhoneSet.silences '(#))
    (provide 'mrpa_phones)

- Lexicons
  - A Lexicon in Festival is a subsystem that provides pronunciations for words.
    - Currently Festival supports a number of different lexicons. They are all defined in the file 'lib/lexicons.scm' (CUVOALD (default), CMU (American English pronunciation on the darpa phoneset) , mrpa, BEEP (British English on mrpa phoneset)
  - It can consist of three distinct parts:

- an addenda, typically short consisting of hand added words
- a compiled lexicon, typically large (10,000s of words) which sits on disk somewhere;
  - It is a file (for example oald-0.4.out) that is produced by using the function `lex.compile`
  - lex.compile takes two filename arguments, a file name containing a list of lexical entries and an output file where the compiled lexicon will be saved.
- and a method for dealing with words not in either list (**letter to sound rules LTS**).

- Lookup process
  - When looking up a word, a word is identified by its headword and part of speech. If no part of speech is specified, nil is assumed which matches any part of speech tag.
  - The lexicon look up process first checks the addenda
  - If no match is found in the addenda, the compiled lexicon, if present, is checked.
    - Compiled lexicons use a binary search method while the addenda is searched linearly
  - Finally if the word is not found in the compiled lexicon it is passed to whatever method is defined for unknown words. This is most likely a letter to sound module

- UniSyn synthesizer
  - It is a waveform synthesizer that is designed to replace the older diphone synthesizer, a single processing module that could be used even when the units being concatenated are not diphones.
  - Diphone names are constructed for each phone-phone pair in the Segment relation in an utterance
  - It uses a database that consists of a set of waveform (may be translated into a set of coefficients), a set of pitchmarks and an index
  - The Unisyn synthesis modules can use databases in two basic formats,
    - separate: when all files (signal, pitchmark and coefficient files) are accessed individually during synthesis. it is designed to be used during development and debugging.
    - group: a database is collected together into a single special file containing all information necessary for waveform synthesis, it is designed to be used for distribution.
      - The function us_make_group_file will make a group file of the currently selected diphone database, it takes as first parameter he name of the file to save in, An optional second argument is a list which allows specification of how the group file will be saved. it can contain the following features
        - track_file_format: The format for the coefficient files. est_binary (default) or est_ascii
        - sig_file_format: format for the signal parts of the of the database. snd (default) [Sun's Audio format]
        - sig_sample_format: format for the samples in the

signal files. mulaw (default and suitable when the signal files are LPC residual)

- Database declaration
  - A database is declared to the system through the command `us_diphone_init`. This function takes a parameter list of various feature:
    - name: used in selecting it from the current set of loaded databases.
    - index_file: A filename name containing either a diphone index, as descripbed above, or a group file.
    - grouped: Takes the value "true" or "false"
    - coef_dir: The directory containing the coefficients, (LPC or just pitchmarks in the PSOLA case).
    - coef_ext: The extension for coefficient files, typically ".lpc" for LPC file and ".pm" for pitchmark files
    - sig_dir: The directory containing the signal files (residual for LPC, full waveforms for PSOLA).
    - sig_ext: The extension for signal files, typically ".res" for LPC residual files and ".wav" for waveform files.
    - default_diphone: The diphone to be used when the requested one doesn't exist
    - alternates_left: A list of pairs showing the alternate phone names for the left phone in a diphone pair.
    - alternates_right: A list of pairs showing the alternate phone names for the right phone in a diphone pair
  - Example

```
(set! rab_diphone_dir
"/projects/festival/lib/voices/english/rab_diphone")
(set! rab_lpc_group
    (list
     '(name "rab_lpc_group")
     (list 'index_file
         (path-append rab_diphone_dir
"group/rablpc16k.group"))
     '(alternates_left ((i ii) (ll l) (u uu) (i@ ii) (uh @) (a
aa)
                       (u@ uu) (w @) (o oo) (e@ ei) (e ei)
                       (r @)))
     '(alternates_right ((i ii) (ll l) (u uu) (i@ ii)
                        (y i) (uh @) (r @) (w @)))
     '(default_diphone @-@@)
     '(grouped "true")))
(us_dipohone_init rab_lpc_group)
```

  - UniSyn module selection
    - (Parameter.set 'Synth_Method 'UniSyn)
      (Parameter.set 'us_sigpr 'lpc)
      (us_db_select rab_db_name)
- Generating a diphone index
  - The diphone index consists of a short header followed by an ascii list of each diphon. For most databases this files needs to be generated by some database specific script

- Example of Header
  - EST_File index
    DataType ascii
    NumEntries 2005
    IndexName rab_diphone
    EST_Header_End
- The entries in the list may take on one of two forms,
  - full entries: consist of a diphone name, where the phones are separated by "-"; a file name (which is used to index into the pitchmark, LPC and waveform files) and the start, middle (change over point between phones) and end of the phone in the file in seconds of the diphone.
    - r-uh edx_1001 0.225 0.261 0.320
  - index entries: simply states that reference to that diphone should actually be made to another.
    - aa-ll &aa-l
    - //This states that the diphone aa-ll should actually use the diphone aa-l

- Adding festival-tts-arabic-voices
  - My notes
    - Copy languages folder to festival/lib
    - Unzip releases/ara_norm_ziad_hts.zip to festival/lib/voices/arabic/ara_norm_ziad_hts
    - Replace all occurnces (5) of "/usr/share/festival/voices/arabic/ara_norm_ziad_hts/festvox/ara_norm_ziad_char_phone_map.scm" [in ara_norm_ziad_lexicon.scm to your path
    - Copy festival/lib/voices to festival/voices
    - To make festival --tts --language arabic test.txt:
      - In festival/lib/languages.scm:
        - After the definition of spanish language there is the definition of select_language function, replace it with the following code (the code defines the arabic language and re-defines the select_language function)

```
(define (language_arabic)
"(language_arabic)
Set up language parameters for Arabic."

  (voice_ara_norm_ziad_hts)
  (set! male1 voice_ara_norm_ziad_hts)

  (Parameter.set 'Language 'arabic)
)

(define (select_language language)
  (cond
   ((or (equal? language 'britishenglish)
   (equal? language 'english))  ;; we all
know its the *real* English
```

```
   (language_british_english))
  ((equal? language 'americanenglish)
   (language_american_english))
  ((equal? language 'scotsgaelic)
   (language_scots_gaelic))
  ((equal? language 'welsh)
   (language_welsh))
  ((equal? language 'spanish)
   (language_castillian_spanish))
   ((equal? language 'arabic)
   (language_arabic))
  ((equal? language 'klingon)
   (language_klingon))
  (t
   (print "Unsupported language, using
English")
   (language_british_english))))
```

- To test
  - In festival:
    - Set the voice: (voice_ara_norm_ziad_hts)
    - Say something: (SayText "السلام عليكم")
    - Say file: (tts "test.txt")