Task 1: Processing transcription-from-linguists.txt.

Instructions:

#1. Merge any consecutive symbols other than phoneme symbols (in 1st column forPhonemeMapping.txt and timitIPAtoArpa.txt) and "sil" phonemes into one.
#2. Symbols in the source file are in IPA format, these need to convert into ARPA format using the files "forPhonemeMapping.txt" and "timitIPAtoArpa.txt"
    #1. in "forPhonemeMapping.txt" first column is ipa and third column is arpa
    #2. in "timitIPAtoArpa.txt" first column is ipa and second column is arpa
#3. Remove all punctuation marks other than "~". Make sure one space before and after "~" in all. Remove consecutive spaces. Arrange the transcriptions in the ascending order of the file numbers
#4. Using "myphones.60-48-40.map", map the symbols in the first column with the symbols in the third column.
#5. Using second and third columns in the file "subMapInfo.txt" replace the file names.
#6. Make sure one entry for one filename. Arrange the transcriptions in the ascending order of the file numbers
#8. Make sure the number of spaces in the text transcriptions should be the same as the number of ~ in the phoneme transcriptions.

Task-1: Part-1 + Part-2 + Part-3 + Part-4 +Part-5 +Part-6

Task-1: Input - “Transcriptions_from_linguists.txt”, Output - “txt file containing unique fileNames along with their mapped phoneme transcriptions from IPA to ARPA, in which the phoneme set is only 40.”

Part-1 (a single code should give this output): Input1 - “Transcriptions_from_linguists.txt”, Output1 - “Unique_Transcriptions_from_linguists.txt” .
“Unique_Transcriptions_from_linguists.txt” = “txt file containing unique fileNames based on latest time stamps along with their phonetic transcriptions”
Verification_code: Check if the created file-phonetic transcription pair exists in transcription_from_linguits.txt using regex to match file-phonetic transcription pattern in Transcriptions_from_linguists.txt.

Part-2 (a single code should give this output): Input2 - “Unique_Transcriptions_from_linguists.txt”, Output2 - “unique_phoneme_symbols.txt”
“Unique_phoneme_symbols.txt” = “txt file containing all phoneme symbols in the input file”
Verification_code: checking if total phoneme symbols contain all characters in the document by comparing total length of total phoneme symbols to length of all characters in the document excluding space, “~” and “?”,”!”,”,”,”;”

Part-3 (a single code should give this output): Input3 - “Unique_phoneme_symbols.txt”, “forPhoneMapping.txt”, “timitIPAtoArpa.txt” Output3 - “Invalid_phoneme_symbols.txt”
“Invalid_phoneme_symbols.txt” = “txt file containing symbols that are not in the following two files -- forPhonemeMapping.txt, timitIPAtoArpa.txt”
Verification_code: Match the symbols contained in Invalid_phoneme_symbols.txt to the text in Unique_Transcriptions_from_linguists.txt to confirm that they all exist.

Part 4- Input: “Invalid_phoneme_symbols.txt”, “Unique_Transcriptions_from_linguists.txt”, “forPhoneMapping.txt”, “timitIPAtoArpa.txt”, Output4:
“IPAtoArpa_converted_transcription.txt”=“consecutive spaces “-” converted to consecutive “~” , Punctuation removed, space before and after “~” maintained, invalid converted to valid and IPA to ARPA converted phonetic transcription”
Verification: Check if all the converted phonetic symbols exist in Col 3 of forPhoneMapping or col 2 of timitIPAtoArpa.

Part 5- Input5: “IPAtoArpa_converted_transcription.txt”, “myphones.60-48-40.map” Output5: “Arpa_to_myphonesmapped.txt”=“ ARPA symbols converted to phoneme set given in myphones.60-48-40.map.”
Verification: check if the converted text units exist in col 3 of myphones.60-48-40.map

Part 6- Input6: “Arpa_to_myphonesmapped.txt”, “subMapInfo.txt”
Output6:”Transcription_from_linguits_processed”=”Transcriptions arranged in ascending file number and file number mapped into respective label given in subMapInfo.txt”

Objectives:

1. From the file (“transcription-from-linguists.txt”) containing date,time, file number, english sentences and phoneme symbols, convert phoneme symbols corresponding to each file from IPA to ARPA format using “forPhonemeMapping.txt” and “timitIPAtoArpa.txt”
2. Multiple consecutive “-” in sentences should be equal to multiple consecutive “~” in phonetic transcription.
3. Remove punctuations after conversion. Add spaces before and after “~” if there isn’t. Remove consecutive spaces, if any.
4. Map the converted ARPA format symbols into conversions given in “myphones.60-48-40.map”
5. Obtain only one file and one corresponding transcription for each file. Arrange them in ascending order based on file number.
6. Replace file names into using data provided in “subMapInfo.txt”

Interpretation/Understanding:

1. Find part of the text which contains phonemes for each file and convert phoneme symbols which are present in either column 1 of “forPhoneMapping.txt” or “timitIPAtoArpa.txt” to column 3 in the former and column 2 in the latter file.
2. For multiple “-” in english transcription , add multiple “~” in phoneme symbols.

3. After conversion, remove any punctuation from the transcription. Provide one space before and after “~” and replace any multiple consecutive spaces with only a single space.
4. Map the converted further using “myphones.60-48-40.map”, using col 1 and col 3 as key-value pairs.
5. Consider a file, only once and correspond/correlate it to a single transcription of converted symbols. Arrange such correlated combinations in ascending order based on file number.
6. Map file names to corresponding labels according to “subMapInfo.txt” using col 2 and col 3 for key-value pairs.

Expected Outcome:

1. Total number of files and their transcriptions come out to be 2927. Symbols present in column 1 are converted to respective symbols in column 3 in forPhonemeMapping.txt and timitIPAtoArpa.txt
1. Places with “--” replaced with just “~ ~” in phonetic transcription because “--” are present in both sentences and transcriptions. (there are two “-” at max).
2. No punctuation symbols present. One space before and after “~” and no multiple consecutive spaces.
3. Processed Symbols in text which are present in column 1 are converted to column 3 from “myphones.60-48-40.map”
4. Each file taken only once results in sorted pairs of file names and converted phonetic transcription come out to be 2442 in number.
5. File names replaced by respective Intonation/Phoneme/Stress/Sentence.

Steps for Algorithm:

1. In lines of text in “transcription-from-linguists.txt” corresponding to each file, obtain part of text containing file number, sentences and phonetic transcription separately. This is done by finding matches to patterns of text (using regex) corresponding file names, english words and phonetic symbols.
   Delimiter for beginning string of phoneme symbols: “.,”, ”?,”, “?.,”, “..,”, “).,”, “!.,”
   Delimiters for ending of string: “?\n”, “!\n”, “.\n”, “)\n”,”\n”
2. Extract all symbols from the transcriptions_from_linguistis.txt, check if those symbols all exist in given ‘phoneMapping.txt’ & ‘timitIPAtoARPA.txt’ files. If they don’t, create a set of those symbols and compile into a text file to send for verification.
3. Create key,value pairs of col 1 and col 3 from “forPhonemeMapping.txt” and “timitIPAtoArpa.txt” for conversion.
4. For each line in stored phoneme symbol transcription, process phonetic symbols in sets of 3 first, (ex- d̪h, t̪h etc), followed by set of 2 (ex- tʃ, ao) and then set of one. Sets are matched using a regex pattern. Order is important so that a part of collective phonetic symbol isn’t converted as a single phoneme symbols (ex: t/t̪ will be converted alone in the set of 3 or of 2 phonetic symbol).

5. “--” replaced “~ ~” in phoneme symbol transcription. Punctuations (‘,’ , ‘.’, ‘!’ , ‘?’ etc) replaced with space.
6. Converted phoneme symbols further processed in sets of 3 and sets of 2 using key-value pairs of col 1 and col 3 in “myphones.64-48-40.map”. (single phoneme symbol is mapped to itself therefore that is left)
7. File names and processed phonetic symbols are zipped together. Key, value pairs of file and processed phoneme symbols are created while iterating through zip, hence storing the single unique pair.
8. File names are sorted separately in another list and iterated through, the corresponding transcription which is saved as a value in the dictionary in point 6, which maps file names and transcription, is used to access corresponding values. This way, transcriptions and files are sorted and stored in a string.
9. File names are replaced by respective Intonation/Stress/Phoneme/Sentence by using key-value pairs of second and third column in “sybMapInfo.txt”

Verification:

1. To ensure if the correct number of separate matches for patterns of files and phoneme symbol transcription have been matched, their total number of occurrences are compared. Later, they’re zipped together so a file corresponds to their respective transcription.
2. To avoid conversions which are invalid, obtain symbols before converting which are not spaced properly, clumped with other symbols or not in the first column of “forPhoneMapping.txt” or “timitIPAtoArpa.txt”. Set of phoneme symbols which are greater than 3 is checked. Sets of 3,2 and single phoneme symbol are checked if it does not belong to the column 1 of above stated files. The matches bring symbols improperly spaced or invalid.
3. If they’re improperly spaced, they’re spaced accordingly by breaking clumped phonemes and space is added in the middle (in sets of 3 or more, manual intervention may be required for replacing). The invalid ones are replaced with manually corrected versions.
4. After conversion, converted phonemes are checked if they exceed a set of 3 or if sets of 3,2 are in column 3 of “myphones.64-48-40.map”. If they do not exceed 3 and there is no case of converted symbol not found in myphones.64-48-40.map, conversion was apt.

TASK 2: Processing of transcriptions based on the files 'sharmistha-output_set1.txt' and 'siddharth-output_set1.txt'.

Instructions:

Follow the same instructions as TASK 1, in addition, follow the below instructions:

#1. Multiple entries for a single file entry then take the one which is the latest based time entry.

#2. Make sure number spaces in the text transcriptions should be the same as the number of ~ in the phoneme transcriptions. The files which are not obeyed, please separate them.
The files which are not obeyed, separate them.
#3 Compare your processed output file with the attached groundTruth file on the entries common in both the files.

TASK 2, PART 1: Maintain same format as task 1 text file by required changes/conversions of delimiters
PART 2: Use task 1, part 2 code to get unique phoneme symbols
Comparison Task: Compare out file of task 2, part 1 and task 1, part 1 for common files and see if they're same.

Objectives:
Same as TASK 1 with the addition:

1. In case of repeated entries, consider the last entry.
2. The files which have multiple "~" in phonetic transcription when there is no corresponding multiple "-" in english sentences are to be separated. For the rest, Multiple consecutive "-" to be converted into the same number of consecutive "~" in phonetic transcription.
3. Find differences compared to the groundTruth file.

Interpretation:
For additional instructions:

1. In case of repeated transcriptions, use the last occurence.
2. If "--" not in english sentences, but "~ ~" in phonetic transcription, those files are to be separated. For others, "--" to be replaced in phonetic transcription with just "~ ~".

Expected Outcome:

1. Total combinations of file and their phonetic transcription come out to be 3646. Only one pair of file name and phonetic transcription formed for each file, where the last occurrence of the pair is considered. This results in the total file-transcription combination for each file to be 3024.
2. Files separated where phonetic transcription has "~ ~" but no corresponding "--" in sentences. For the rest, the "--" in sentences converted to "~ ~" in phonetic transcription.

Steps for Algorithm:
Steps 1,2,3,5 as mentioned in TASK 1 Algorithm. Additional steps:

1. English sentences and their phonetic transcription are zipped together. Iterating through both of them together, it is checked if phonetic transcriptions have "~ ~" but english sentences don't have "--" then they are separated.
2. Iterating through zip, if the sentences don't have "--" but phonetic transcriptions have "~ ~" or "--", those phonetic transcriptions are stored and removed. For others, the "--" is replaced with "~ ~".

3. File names and processed phoneme symbols are zipped together. Key, value pairs of file and processed phoneme symbols are created while iterating through zip, hence storing the last unique value of a pair. Pairs arranged by sorting file names in ascending order and corresponding transcription is also ordered accordingly.
4. Check the number of files and their transcription in groundTruth and produced file to ensure the same numbers. Store differences by creating sets of both files and subtracting them. Separate intonation/Phoneme/stress/Sentence and their transcription and zip them together. Iterate through files which are present in the produced file and check for characters stored in differences if they're present and in groundTruth file, get the respective intonation/stress/phoneme and transcription. Check for the same but in the produced file next time. This way, it is known which differences are present in one file and absent in another.

Verification:

Steps same as TASK 1 along with the addition:

1. After replacing respective "--" in phonetic transcription with "~ ~" , the processed phonetic transcription lines are zipped with respective sentences and iterated through simultaneously to check if sentences have "--" and phonetic transcription does not have "~ ~". If none appears, the change has been made accordingly.

TASK 3: Processing files in Drive.

Part 0:
TASK 3, PART 1: Convert into text file and make unique files document
Part 2: Using task 1, part 2: Get all unique phone symbols in task 3 files

Instructions:

#1. Get the unique files with the transcriptions based on "file_XXXX". The files which are repeated in this list, choose the entry with the latest timestamp. After this you can expect 3236 entries in the file.
#2. Map the phonemes as earlier.
#3. Map the "file_XXXX" using submapInfo.txt file considering subject name and intonation/phoneme/sentence/stress file no. After this you can expect 3236 entries in the file. Also, in this file you can observe the entries with file names of the format "name Intonation/Phoneme/Sentence" have repetitions at max two.
#4. Split the mapped file into two. One file with one entry corresponding to each file name. Second file with other entries of the repeated file names. You can observe here the sum of the total number of entries in both the files equal to 3236.
#5 Remove the spaces and insert "_" between name and intonation/phoneme/Stress/Sentence

Objectives:

1. Obtain files and their transcription exactly one time for each file.

2. Map phonemes using “forPhoneMapping.txt”, “timitIPAtoArpa” and “myphones.64-48-40.map”.
3. While mapping file names using “subMapInfo.txt” use column 1 entries along with column 2 for replacement with their corresponding value in column 3.
4. Create two mapped files, one with mapped file names and their respective transcription for files, taking their occurrence only once. Other one having remaining entries of the mapped files which are repeated.

Interpretation:

1. Take each file and their transcription exactly one time, using their last occurrence.
2. Map phonemes using steps in Task 1.
3. Map file names including sub name in column one of “subMapInfo.txt”, remove spaces between sub name and intonation so it’s converted in the format of “name_Intonation/Phoneme/Stress/Sentence”
4. In the file containing all mapped file names and their transcription, create two files, one with each mapped file and it’s transcription once. Other, containing remaining repetitions of mapped file, along with its processed transcription.

Expected Outcome:

1. Taking each file exactly once results in total files and their transcriptions to be 3236.
2. Processed files and their transcription forms the format of : “name_Intonation/Phoneme/Stress/Sentence” “converted transcription”.
3. First map file contains all files and their transcription only once, resulting in a total of 3024 such files. The one containing the remaining repetition comes out to be 202 in number.

Steps for Algorithm:

Common steps as mentioned in Task 1.

1. Pre-Processing: Iterate through all file names in the folder and convert them from docx to txt using docx2txt, convert rtf file to txt file.
2. Make key-value pairs of subMapInfo.txt where key is file name (col 3) and value is sub name_Intonation (col1+col2)
3. Make a list containing files and their transcription which are repeating (X), based on count of elements (any element has max of 3 counts). Make another list for files which are taken only once (Y) . Pop elements in repeating list (X) if they are present in Y.

Verification:

Verification of correct conversion/processing as mentioned in task 1 and task 2.Additional:

1. The number of file-transcription combinations in split file 1 should be 3024 and In second file, 202, so their sum total comes out to be 3236.

TARGETS FOR NEXT MEET:

1. Fix the i : issues and re run part 3, send output files
2. After comparison of shar_sid and transcription from linguist files, make changes in either of the files based on the difference.

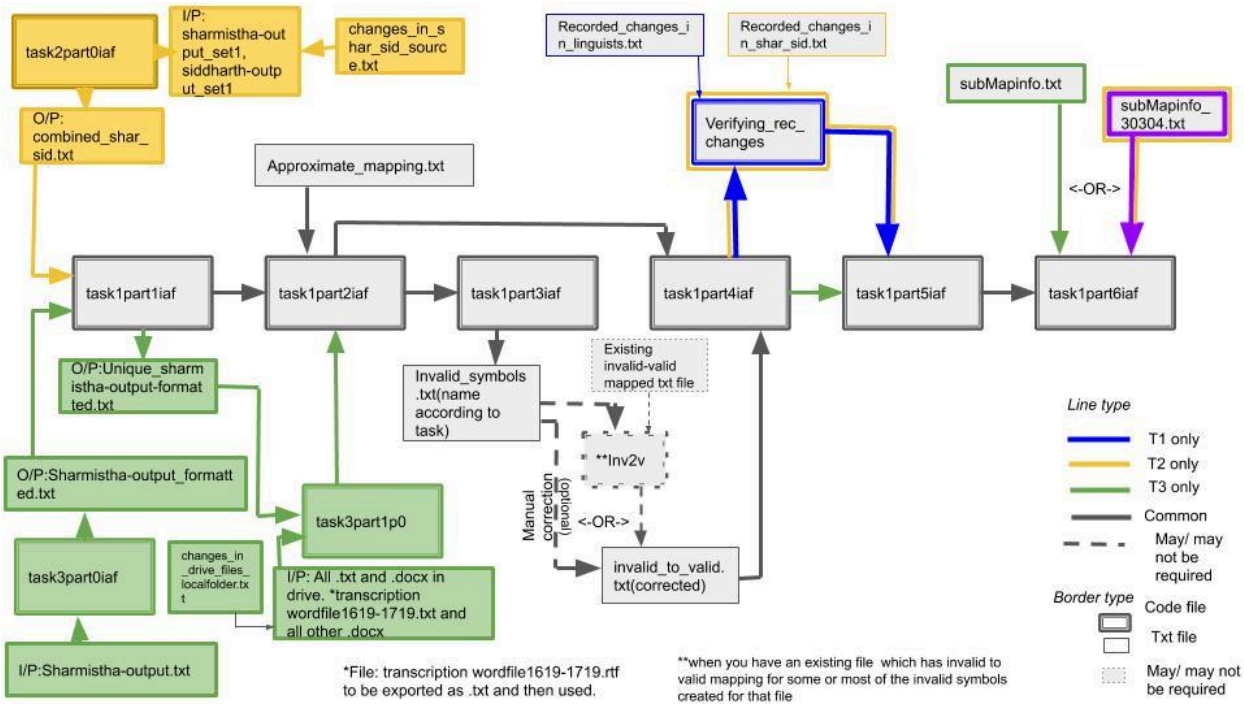
3. Json files sample send
4.

Current target:

The dH AX 45
The dH IX 36
format= word \t ipa trans \t occurence of that word with the ipa transcription (in the whole doc) \t arpabet trans \t occurence of that word with arpa transcription (in the whole doc)

-> send package of all tasks wrapped
-> recorded changes (logs)

Changes made:
Pipeline flow:



Major changes in codes:

Common across most: Added condition for input, to make it work with text file name as input or file as a string itself.

1. Task 1, part 1(task1part1iaf):
a. Time stamps compared as int types: tsdigits.append(int(ts_str))

   b. Wrapped in a function named part1()
2. Task 1, part 2(p2new):
   a. Added line where all non breaking spaces and special spaces are replaced with a common space. (i.e utfl=re.sub(u'[\xc2\xa0\u2000\u2001\u2002\u2003\u2004\u2005\u2006\u2007\u2008\u2009\u200A\u2028\u2029\u202F\u205F\u3000]',' ', utfl) #replacing non breaking spaces with space)
3. Task 1, part 4(p4p4):
   a. Added conditions for replacement, defining how replacement should work when there is space only before invalid symbol(ending in a sentence), space only after symbol (beginning of a sentence) or when there is space before and after (mid) or when there's space before and after.

      ```
      if re.search(r'^[^\s~]+',p).group(0)==j: #if a transcription starts with invalid phonetic symbol
                  p=re.sub(rf'(?<!\D){j}(?= )',correction_itv[j],p) #check if nothing is in front of it and is followed by a space
                  if re.search(r'[^\s]+$',p).group(0)==j: #if invalid symbol is a the end of a transcription line
                  p=re.sub(rf'{j}',correction_itv[j],p) #replace it with corrected symbol (space may or may not be there before it)
                  else:
                  p=re.sub(rf'(?<= ){j}(?= )',correction_itv[j],p) #else check for space before and after then only replace it.
      ```

   b. Removing space at the beginning of string, end of string and replacing multiple spaces with single space.
      ```
      p=re.sub(r'^ +','',p)
      p=re.sub(r' +$','',p)
      p=re.sub(r' +',' ',p)
      ```