Python for NLP

Final Project

Auromita Mitra

H00MAENG20190259

Project title: Phonetic Word-Finder for Bengali

Introduction:

When designing stimuli and test material for phonetic/psycholinguistic experiments, researchers often need words of a particular length, or containing particular sounds (e.g. monosyllabic words containing /a/, words ending with a nasal, etc). Finding such items through introspection is difficult and time-consuming, even for a native speaker, especially if a large number of words are needed. An automated system to find words from a language with certain phonetic properties could make this task much simpler — even rough estimates of properties like syllable length, presence/absence of a sound group, could narrow down the list of possible words to choose from.

While a website with this kind of functionality exists for English (https://lessonpix.com/SoundFinder -- finds English words having certain sounds in particular positions), there is no such tool available for Bengali. It is not practical to use a dictionary for this because—i) it only allows us to specify the sound in the word-initial position, with no control over word length/other sounds; ii) some vowel letters in Bengali can have multiple realizations, and thus searching by letters is not adequate to get the required sound (e.g. অ can be realized as either /ɔ/ and /o/, depending on the word); iii) often, words satisfying multiple criteria are required—dictionaries cannot provide such information.

Objective:

To create a program which generates a list of legitimate Bengali words fulfilling certain phonetic criteria, as specified by the user. This includes the number of syllables, and sounds in specific positions (beginning, middle, end). The focus will be to make the program usable for linguistic research (e.g. by classifying sounds into linguistically meaningful groups—nasals, plosives, retroflex, etc), and user-friendly (require minimum typing, no IPA symbols, diacritics etc).

Program Folder Documentation:

The folder contains the main program (bengali_word_finder.py), four supporting modules (data, sounds, subsets, display), a phonetic dictionary (shruti.dic) and an output file (output.txt). An overview of their contents is given below. Detailed documentation about the modules can be found by invoking the help() function. These form the three main components of the program:

A. Data component

i) shruti.dic

This is the phonetic dictionary from the SHRUTI corpus. It has been processed to be usable as a python dictionary object. It contains 22012 words (transliterated in the ITRANS format) and their phonetic transcriptions (following the system documented in Das et al.(2011)).

ii) data.py

Contains the data used in the program, and the functions for data processing. Since this program performs all operations on the sound features, the transcriptions are kept as the dictionary keys, and the corresponding words as the values.

iii) sounds.py

Contains lists for all sounds and sound groups used in the program.

B. Core component

iv) subsets.py

Contains all the functions used for detecting sound and syllable information using the classifications in sounds.py, and generating lists of words fulfilling each condition. These are all generic functions, and can be used on any input data of similar format.

C. Interactive component

v) display.py

Contains all the display strings used in the interactive parts of the program. These were kept in a separate module to avoid cluttering the main program code, and also make them easier to format/edit.

vi) bengali_word_finder.py

Contains functions which allow (a) the core functions to interact with each other; (b) the program to interact with the user. All of these use the input() function to get user input, and control structures to respond accordingly. The interactive part is divided into components (called "pages" for convenience), which are arranged in a hierarchical order (details in Usage section). Each page is associated with a single function—"pagename"_func(), which uses combinations of core functions to carry out its specific objective. 'Going to a page' amounts to invoking the page function. After completing the page-specific objective, it allows the user to directly invoke another page function (go to another page). Thus, it is possible to move between pages while the program is running. Doing this does not erase the data created during the session. Running the same function again overwrites the earlier data generated by it—thus, it is possible to modify parameters (e.g. add or remove conditions) within a single session. On exiting the program, all data is erased.

The page_func() function acts as a central redirecting function—it reads the user input and invokes the appropriate function. This is done so that separate control structures are not needed for every possible input. Similarly, the home_func() serves as a central "page" from which any of the other functions can be accessed.

Most of the interactive functions have an "error-clause" which displays an error message if the user input is unexpected, and invokes the same function again. This is done to prevent the program from crashing due to errors in the input command (such as spelling mistakes or case-mismatch).

The main program is a single-line command which invokes the HOME page. After that, the functions keep redirecting among themselves to generate, modify, and save data. The program closes when prompted by the user.

Corpus and transcription information:

Data is taken from the SHRUTI corpus (a read Bengali speech corpus designed by the Indian Institute of Technology, Kharagpur, freely available for non-commercial use). It contains 7383 unique sentences from Bengali newspapers in different domains, covering the most frequently used words of the language, and a phonetic dictionary based on the utterances, which contains 22012 words and their phonetic transcriptions. The words are transliterated using the ITRANS format (Indian languages TRANSliteration – an ASCII transliteration scheme for Indic scripts)

Phonetic transcription in the program follows the system described in Das et al.(2011). Apart from two special characters (^ and ~), it uses only alphanumeric characters. The system uses both uppercase and lowercase letters. Therefore, the input commands in the program are case-sensitive. Das et al.(2011) identify 47 phonemes based on automatic speech recognition of the corpus data. These are listed below:

Consonants: [k kh g gh ch chh j jh ^n Y sh T Th D Dh rr R tt tth dd ddh n l s p ph b bh m h]

Vowels: [i ^i ~i ee ^ee E ^E A ^A a oh ^oh u ^u oi ou ^ou aa ^aa]

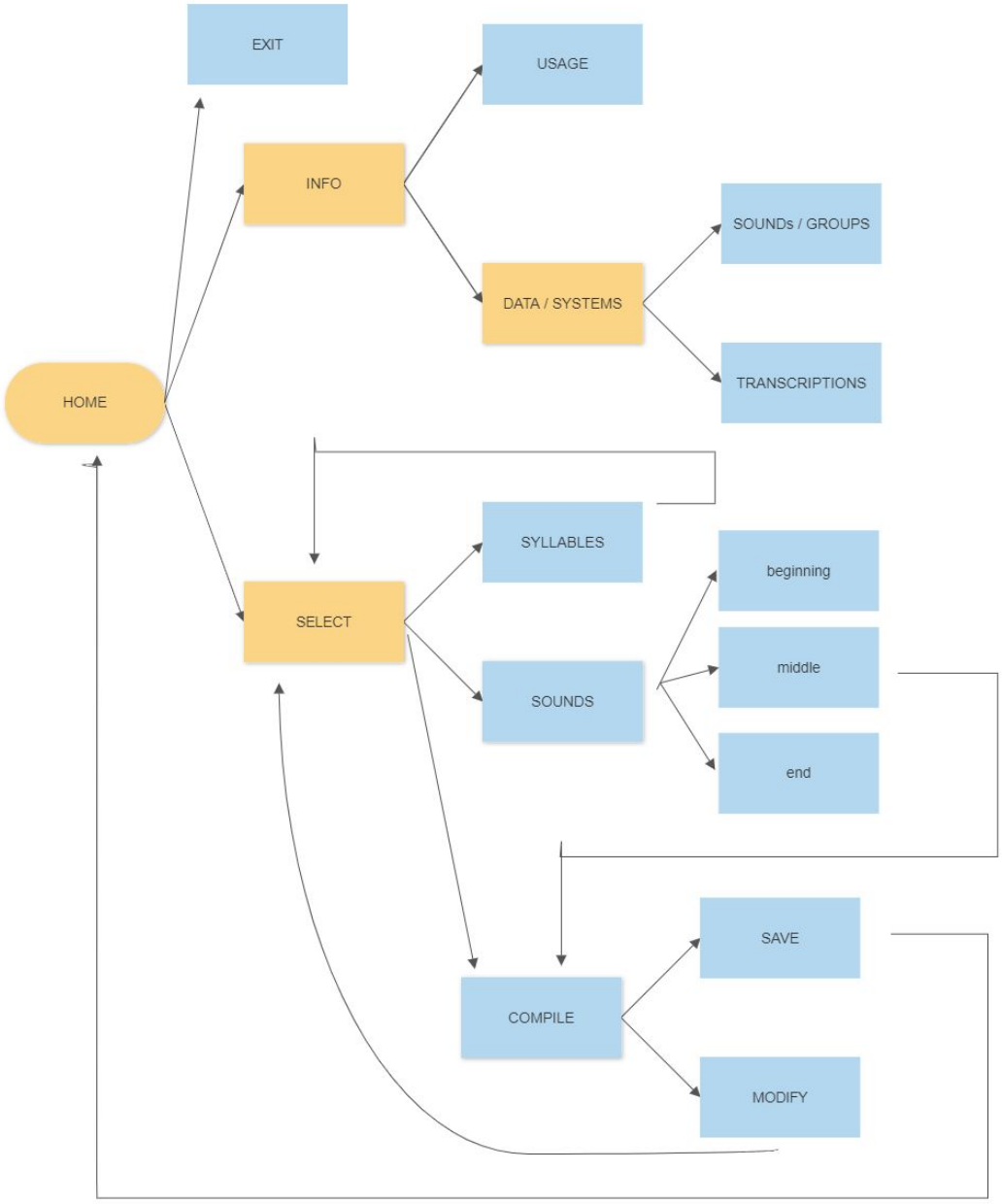
The following sound groups are specified:

voiced, voiceless, aspirated, nasals, fricatives, stops, semi-vowels, labial, denti-alveolar, retroflex, postalveolar, velar

Since these are not unambiguous (group membership of certain sounds are debated in the literature), the complete list of groups and sounds are provided in the INFO page for the convenience of the user.

Usage:

The sequence of pages is given below. HOME can be accessed from any page. Details regarding the input commands are added in the INFO page.



i) SYLLABLE S and SOUNDS can be modified any number of times. Each modification overwrites previous selections. To specify multiple conditions (words matching any of the specified conditiona are allowed), separate them by commas.

ii) If the user does not want to specify any constraints on a given parameter, they can type NONE.

iii) The COMPILE page allows user to see the list generated by the current selection. If user is not satisfied, they can go back and select certain parameters again, without affecting others (selecting SOUNDS preserves earlier SYLLABLES selection). SAVE creates an output .txt file. An example output file with the specification SYLLABLE-1, SOUNDS: beginning—"stop", middle—"A", end—"None" is included in the program folder (example.txt).

Limitations and possible extensions:

i) Although error clauses are added to the interactive functions to prevent the program from crashing due to input error, some processes feed this input to the core functions. Right now, the core functions are not equipped to handle unexpected input. Thus, errors in certain inputs might cause a crash. To minimize the chances of this, options are displayed with each input prompt.

ii) Right now, the program uses a rough measure of 'syllable' (as no. of vowels). While this gives fairly accurate results for the number of syllables, it is not possible to define syllable boundaries. Thus, it is not possible to specify sounds in particular syllable positions (e.g. a nasal in the first syllable). Enriching the data with more detailed syllabification rules for Bengali will make this possible.

iii) The current program can only specify positive values. Although the subsets module contains a function notlist() for creating complement lists for single conditions (e.g. list of all the words which don't begin with a vowel), right now there are no control structures which allows this to inerct with the positive functions through AND/OR operations in the interactive part.

iv) Similarly, it will be useful to add a function which will allow the user to specify ranges of syllable length (e.g. < 5 – right now, the user must write this as 1,2,3,4)

v) Since the corpus contains transcription files, its possible to get frequency information for the words. Since word frequency is often used as a factor to select experiment stimuli, listing frequency information in the dictionary, and allowing the user to filter by frequency will make the program more useful. I didn't get time to try this, but I want to add this in the future if possible.

References

Das, B., Mandal, S., & Mitra, P. (2011, October). Bengali speech corpus for continuous auutomatic speech recognition system. In *2011 International conference on speech database and assessments (Oriental COCOSDA)* (pp. 51-55). IEEE.