Documentation: 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.

The word-finder uses an underlying corpus to generate lists of words matching specified phonological descriptions. These can be the presence or absence of certain sounds at given positions, or number of syllables. Apart from single sounds, it also pre-defines linguistically relevant sound groups, so that it is possible to find, for example, words that begin with a nasal, or contain a voiced retroflex. It is also possible to preview, edit, and filter a current selection before generating an output file with the list of words.

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 Python functions for data processing. Since this program performs all operations on the sound features, the transcriptions serve as dictionary keys, with the corresponding words as its value.

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 looking up words in shruti.dic that fulfill 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), so that 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 can keep redirecting among themselves to generate, modify, and save data. The program closes when prompted by the user.

Usage:

- Running the program opens up the HOME page. From here, either

  (i) go to INFO for information about commands, available sound groups, and usage instructions; or
  (ii) go to SELECT to start creating a wordlist

- The SELECT page allows you to specify two properties:

  (i) SYLLABLES: enter the desired number of syllables. To specify a range, type each number in the range (e.g. 2 to 5 syllables translates to *2,3,4,5*). Go back to SELECT.
  (ii) SOUNDS: enter conditions for sounds at the beginning, middle, and end of the word. This can be a specific sound, or any of the pre-defined sound groups. Separate multiple condition with commas (e.g. *BEGINNING: retroflex,voiced* – outputs words starting with voiced retroflex sounds)

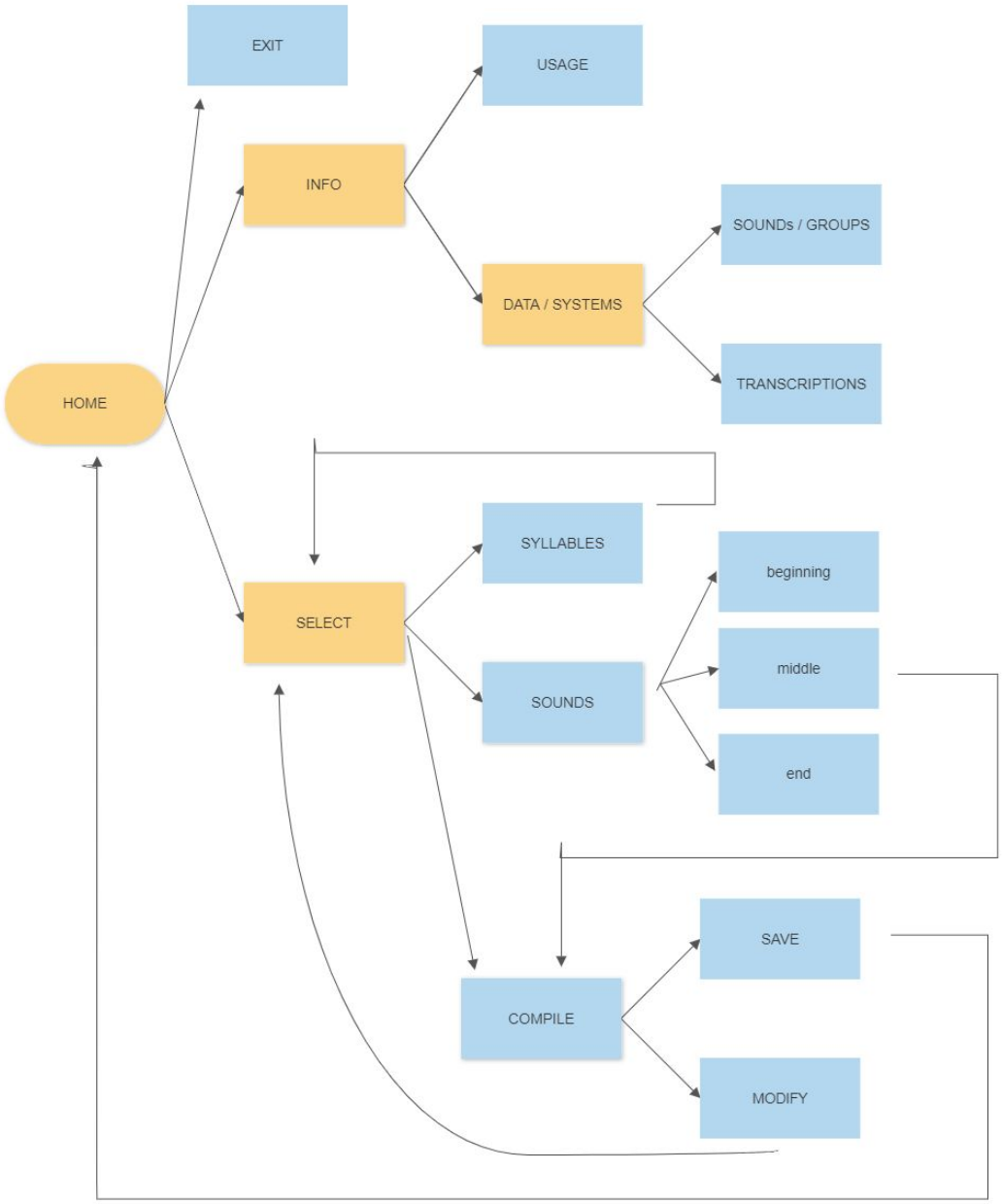
  note: Syllables and sounds can be modified any number of times. Each modification overwrites previous selections. To specify no constraints on a given parameter, type NONE (e.g. SYLLABLES: NONE matches words of any length that fulfill the other conditions).

- After specifying sounds and syllables, go back to HOME. Now, the COMPILE page shows the list generated by the current selection. To make any changes, you can go back and specify certain parameters again, without affecting the others (modifying SOUNDS preserves earlier SYLLABLES selection).

- SAVE creates an output text file with the list of words (transliterated using ITRANS) and their phonetic transcriptions.

An example output file with the specification SYLLABLE-1, SOUNDS: beginning—"stop", middle—"A", end—"None" is included in the program folder (example.txt).

The figure below is a visualization of how pages are structured. HOME can be accessed from any page. Details regarding input commands can be found in the INFO page of the program.



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 is provided on the INFO page inside the program.

Some known issues:

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.py 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 interact with the positive functions through AND/OR operations.

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.