2. Take Hindi (or your mother tongue) and suggest how the code that you wrote in point 1 will have to changed to

work for Hindi. Can it work with same accuracies, or will there be problems specific to Hindi which don’t occur in

English? Explain with examples.

First of all working with Hindi characters is extremely difficult.

This is because we would need to work with UTF8 which can have some rather interesting side effects. Its better that we work with the transliterated Devanagari system called the w-x notation as shown here.

<http://trigonaminima.github.io/2019/03/wx_notation/>

Hindi naturally lends itself to an alphasyllabary in which syllables consist of {CV}+V? in regex format where C is consonant and V is vowel. Thus the porter stemmer analysis is mostly applicable to hindi as well.

Now for the actual stemming of Hindi. Hindi is an extremely inflectional language. Moreover, unlike English, while there are many more inflections to consider, these inflections happen to be rather well behaved. Unlike some other indian languages, the noun inflections are written as separate words simplifying our task.

Eg ladZke kA has a seperate ka in hindi but

mulAcA has a conjoined cA genitive inflection in marathi

Thus the algorithm should first concern itself with the removal of verb and noun grammatical mutations for gender, number, case, person and tense. This would be added as step6

A few examples of this are



Removal of inflections can be shown as

karumgA -> kar

pEsE -> pEs

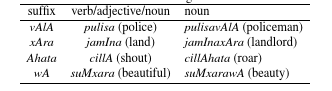
Etc...

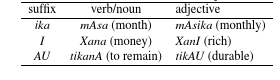
Besides these some irregular plurals and tense mutations will have to be hardcoded. For example in Marathi when certain nouns with long vowels on first syllable are pluralized, the long vowel becomes short. The way to normalize this would be to take the singular form of the noun with the long vowel as the stem.

Example: PUl == Flower but Pule == flowers (U->u)

Here we should take PUl as the stem for both

After that we have the secondary suffixes which are used to nominalize or adjectivize etc the word.We can code in equally effective rules for these suffixes.





Again as these suffixes are rather regular we can simply match the longest possible suffix and remove it from the word

tikAU - > tik

However sometimes hindi mutates the whole word (still with a certain degree of regularity)

praWam .... prAWamik

Here the vovel in the first syllable changes to accomodate the change to an adjective We must consider a seperate scheme for these words. It is not acceptable to treat them simply as exceptions because the changes follow rules of sanskrit grammar and thus can be analyzed however, such an algorithm will be longer than typical implementations of porter stemmer