**Mihir Haresh Shah**

**60004180051**

**BE A/A3**

**Subject: NLP**

**Experiment No:5**

**Aim:** Perform Morphological Analysis on a word.

**Theory:**

Morphemes are considered as smallest meaningful units of language. These morphemes can either be a root word(play) or affix(-ed). Combination of these morphemes is called morphological process. So, word "played" is made out of 2 morphemes "play" and "-ed". Thus finding all parts of a word(morphemes) and thus describing properties of a word is called "Morphological Analysis". For example, "played" has information verb "play" and "past tense", so given word is past tense form of verb "play".

A morphological analyzer is a program that analyzes the morphology of an input word. It uses rules to identify the root and grammatical features of given words. It splits a given word into it's root, lexical category, gender, number, person, case, case marker or tense aspect modality(TAM), suffix and other required grammatical features as given below.

1. root : Root of the word (e.g. ladZake word has root ladZakA)

2. cat : Category of the word (e.g. Noun=n, Pronoun=pn, Adjective=adj, verb=v, adverb=adv post-position=psp and avvya=avy)

3. gen : Gender of the 4. num : Number of the word (e.g. Singular=sg, Plural=pl, dual, and any )

5. per : Person of the word (e.g. 1st Person=1, 2nd Person=2, 3rd Person=3, and any)

6. case : Case of the word (e.g. direct=d, oblique=o and any)

7. tam : Case marker for noun or Tense Aspect Mood(TAM) for verb of the word

8. suff : Suffix of the word

**Analysis of a word :**

बच्चों (bachchoM) = बच्चा(bachchaa)*(root)* + ओं(oM)*(suffix)*  
(ओं=3 plural oblique)  
A *linguistic paradigm* is the complete set of variants of a given lexeme. These variants can be classified according to shared inflectional categories (eg: number, case etc) and arranged into tables.

**Paradigm for बच्चा**

|  |  |  |
| --- | --- | --- |
| **case/num** | **singular** | **plural** |
| **direct** | बच्चा(bachchaa) | बच्चे(bachche) |
| **oblique** | बच्चे(bachche) | बच्चों (bachchoM) |

Algorithm to get बच्चों(bachchoM) from बच्चा(bachchaa)

1. Take Root बच्च(bachch)**आ(aa)**
2. Delete **आ(aa)**
3. output बच्च(bachch)
4. Add **ओं(oM)**to output
5. Return बच्चों (bachchoM)

Therefore आ is deleted and ओं is added to get बच्चों

Add-Delete table for बच्चा

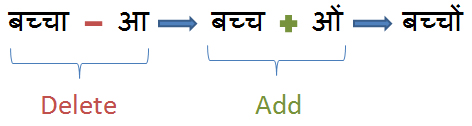
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Delete** | **Add** | **Number** | **Case** | **Variants** |
| आ(aa) | आ(aa) | sing | dr | बच्चा(bachchaa) |
| आ(aa) | ए(e) | plu | dr | बच्चे(bachche) |
| आ(aa) | ए(e) | sing | ob | बच्चे(bachche) |
| आ(aa) | ओं(oM) | plu | ob | बच्चों(bachchoM) |

Paradigm Class

Words in the same paradigm class behave similarly, for Example लड़क is in the same paradigm class as बच्च, so लड़का would behave similarly as बच्चा as they share the same paradigm class.



**Conclusion:**

Words can be analysed morphologically if we know all variants of a given root word. We can use an 'Add-Delete' table for this analysis. 

**Code:**

import codecs

import re

#read the input file

filepath = 'hindi.txt'

f = codecs.open(filepath, encoding = 'utf-8')

text = f.read()

sentences=text.split(u"।") #since hindi sentences end with '|'

words\_list = list()

for sentence in sentences:

    words = sentence.split(' ') #words are seperated by a space in hindi

    words\_list += words

suffixes = {

    1: [u"ाएगी",u"ाएगा",u"ाओगी",u"ाओगे",u"एंगी",u"ेंगी",u"एंगे",u"ेंगे",u"ूंगी",u"ूंगा",u"ातीं",u"नाओं",u"नाएं",u"ताओं",u"ताएं",u"ियाँ",u"ियों",u"ियां"],

    2: [u"ो",u"े",u"ू",u"ु",u"ीय",u"ि",u"ा"],

    3: [u"कर",u"ाओ",u"िए",u"ाई",u"ाए",u"ने",u"नी",u"ना",u"ते",u"ीं",u"ती",u"ता",u"ाँ",u"ां",u"ों",u"ें"],

    4: [u"ाकर",u"ाइए",u"ाईं",u"ाया",u"ेगी",u"ेगा",u"ोगी",u"ोगे",u"ाने",u"ाना",u"ाते",u"ाती",u"ाता",u"तीं",u"ाओं",u"ाएं",u"ुओं",u"ुएं",u"ुआं"],

    5: [u"ाएंगी",u"ाएंगे",u"ाऊंगी",u"ाऊंगा",u"ाइयाँ",u"ाइयों",u"ाइयां"],

}

stems=list()

for word in words\_list:

    for L in range(1,5):

        if len(word) >= L + 1:

            for suffix in suffixes[L]:

                if word.endswith(suffix):

                    word=word[:-L] #stripping the suffix from the word

                    try:

                        if word[-1] == u"ि":

                            word = word[:-1] + u"ी"

                    except:

                        print(word)

    if word:

        stems.append(word)

filename='stems\_generated.txt'

f=codecs.open(filename,"w",encoding='utf-8') #open in write mode

for stem in stems:

    f.write(str(stem))

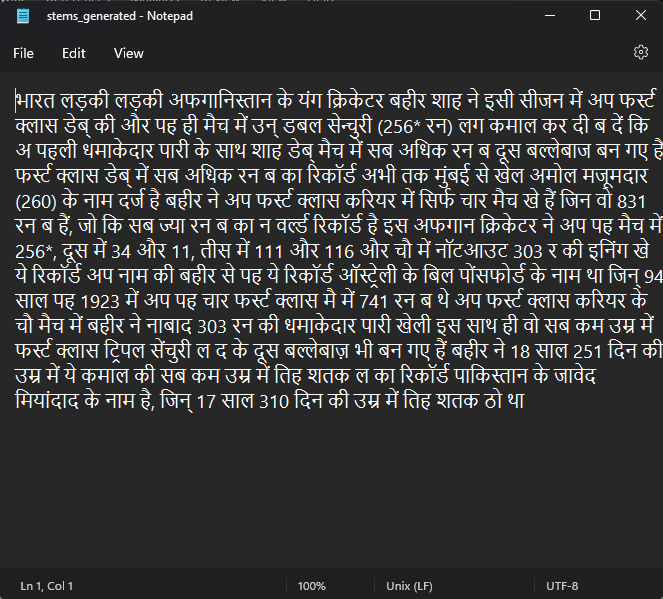
    f.write(u"\u0020")

f.close()

**Text

Description automatically generated**

**Output:**

****

**Conclusion:** Hence, we have successfully performed morphology on a hindi text.