Page No.
Date

Aim- To perform POS tagging using Rule based tagging using

Theory- Pos tagging is the process of making up a word in a corpus to corresponding part of speech tag, based on Rs context & definition. This task is not straight forward as a particular word may have a different part of speech based on the context in which the word is used

The tag in case of is a part of speech tag & signifies whether the word is a noun, adjective, verb & so on.

Parts of speech	Tag
Noun	n
verb	V
Adjective	a
Adverb	r

Rule based POS tagging.

This is one of the oldest technique & tagging. Rule based taggers use dictionary or lexicon for getting possible tag then rule based taggers use hand-written rules to identify the correct tag.

As the name suggest, all such kind of information in rule-based POS tagging is coded in the form of rules

like:- Context pattern rules

- Regex compiled into finite state estimata

Teacher's Sign.: _____

Page No.
Date

Rule based POS tagging is divided into 2 stage architecture

* First stage : It uses a dictionary to assign each word a list of potential POS tags
* Second step - It uses large lists of hand-written disambiguation rules to sort down the list into a single POS tag for each word

Stochastic POS tagging

The model that ~~is~~ includes frequency or probability can be called stochastic. Any no of different approaches to the problem of parts of speech tagging can be referred to as stocastic taggers

Transformation Based Tagging :

It use a rule based algorithm for automative tagging of POS to the given text. TBL allows us to have linguistic knowledge in a readable form, transform one state to or other state by using transformation rules

For the purpose of this experiment, we have chosen the Penn Treebook tagset. This contains 36 to 37 POS tags like NN, VBN, VBD, VBZ, RB, RBD etc

Condusion - Thus we have successfully implemented rule based tagging for given text using the Penn Treebank target.

Teacher's Sign.: _____

Exp 6 : POS tagging

Code :

```
import re
nouns="i,you,bill,back,table,chair,apple,fox,
dog,boy,he,she,it,city,country,car,park".split(
",")
verbs="is,walk,sell,talk,command,belong,try,
understand,love,promised,play,park".split(","
)
preps="of,with,at,from,into,to,in,for,on,by,ab
out,like".split(",")
conj="and,but".split(",")
art="a,an,the".split(",")
pronoun="he,him,his,she,her,hers".split(",")
adj="bad,good,beautiful,handsome,tall,short
".split(",")

def category(root):
    cat=[]
    if root in nouns:
        cat.append("NN")
    if root in verbs:
        cat.append("VB")
    if root in preps:
        cat.append("IN")
    if root in art:
        cat.append("DET")
    if root in conj:
        cat.append("conj")
    if root in pronoun:
        cat.append("PRP")
    if root in adj:
        cat.append("JJ")

    return cat

dict ={}
def tokenize_form(sentence):
    tokens=sentence.split()
    f_tokens=[]
    roots=[]
    for t in tokens:
        w=re.compile(r'[a-zA-Z]+').findall(t)
        #print(w)
        f_tokens.extend(w)
    for w in f_tokens:
      roots.append(w)
      if w.endswith("ing"):
          root=re.sub(r'ing$','',w)
          cat=category(root)
          dict[w]=cat
          print(w,cat)
      else:
          cat=category(w)
          dict[w]=cat
          print(w,cat)

    return roots
string=input("Enter sentence:")
verbs=tokenize_form(string)

import re
# string=input("Enter string to preprocess:")
#tokenisation
tokens=string.split()

#filtration
f_tokens=[]
for t in tokens:
    w=re.compile(r'[a-zA-Z]+').findall(t)
    f_tokens.extend(w)

print("After resolving ambiguity")
#rules
for w in f_tokens:
    if len(dict[w])>1:
        i=f_tokens.index(w)
        cat=dict[f_tokens[i-1]]

        if 'NN' in dict[w] and 'VB' in dict[w]:
            if (cat==['DET']):
                print(w, "NN")
            else:
                print(w, "VB")
```

Output :

```
============================== RESTART: D:/pos.py ==============================
Enter sentence:park the car
park ['NN', 'VB']
the ['DET']
car ['NN']
After resolving ambiguity
park VB
>>>
============================== RESTART: D:/pos.py ==============================
Enter sentence:the boy is playing in the park
the ['DET']
boy ['NN']
is ['VB']
playing ['VB']
in ['IN']
the ['DET']
park ['NN', 'VB']
After resolving ambiguity
park NN
>>>
>>>
```