I J C T A, 9(13) 2016, pp. 6111-6118

An Enhanced Parts of Speech Tagger for English Sentence

Akhil Raj R.[1], Roshmy T. R.[2] and Arun Kumar N[3]

ABSTRACT

In this paper we proposed to introduce an enhanced Parts-Of-Speech (POS) tagging approach for tagging English sentences using statistical approach (n-gram, Hidden Markov Model). As part of implementation, we are using our own Tag Set and a supervised sentence corpus. Most probable tag sequence for the sentence is created by using Rule Based Corpus. We use our own smoothening algorithm to improve the accuracy of the tagged sentences and expand the corpus with newly learned sentences.

Keywords: HMM, Bigram, POS, Tagset

1. INTRODUCTION

Parts of speech (POS) tagging are used for tagging words in English sentence based on the parts of speech grammatical category. POS are mainly classified into 9 categories. POS tagging has great importance in the field of NLP. POS tagging is mainly used in the field of parsing, information retrieval and machine translation. The main problems involved in POS tagging is ambiguity. This leads to some tagging failures. The main approaches used for parts of speech tagging are rule-based, stochastic (n-gram, HMM), and transformation based approaches (Brill's tagger) [1][2].The Stochastic approach make use of HMM (Hidden Markov model) . Here we proposed to introduce an enhanced POS tagging approach to reduce the ambiguity and increase the accuracy of the tagged sentences. As part of this, we make use of our own Tag Set and a stochastic HMM Tagger with Bi-gram assumption [3][4]. Bigram taggers assign tags on the basis of a pair of words [5]. Again we will check the accuracy of the resultant tagged set using our own smoothening algorithm.

2. METHODOLOGY

As part of implementation Hidden Markov Model Bi-gram assumption is used to tag the input sentences. Hidden Markov Model Bi-gram assumption is also known as HMM Model. Penn Treebank and Brown corpus are the commonly used tag set in pos tagging [6].For implementation, we create our own tag set to identify the particular tags we are used in the corpus. These tags are representing an abbreviation of different types of parts of speech used in POS tagging. A supervised sentence corpus [7][8] is created using these tags. This corpus contains different types of tagged sentences. By using these tagged sentences we assign appropriate tags for newly learned sentences with the help of Hidden Markov Model Bi-gram assumption.

Table 1 shows the Tag set used in our implementation.

[1, 2] Dept. of CS&IT, Amrita School of Arts and Sciences, Kochi, Amrita Vishwa Vidhyapeetham

[3] Asst. Professor, Dept. of CS&IT, Amrita School of Arts and Sciences, Kochi, Amrita Vishwa Vidyapeetham

Table 1
Tag set

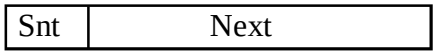
Tag	*Meaning*
AD	Adjective
PR	Preposition
ADV	Adverb
CO	Conjunction
DE	Determiner
NO	Noun
NU	Number
PRO	Pronoun
VE	Verb
IN	Interjections
SENT	Sentence final punctuation

An Example tagged sentence in the supervised corpus is, animals\NO like\PRE kind\AD people\NO #

Each word and its corresponding tag are separated by '\' and each sentence in the supervised corpus ends with '#'.

The process mainly deals with tagging the input sentence by providing high accuracy. This process starts with taking the input sentence. After the user can input the sentence then it will be compared with the sentences in the supervised corpus[9]. At the time of comparison take each sentence from the supervised corpus and remove all tags in the sentence. If the input sentence is similar to one of the corpus sentence then retrieve the corresponding tagged sentence from the supervised corpus and display the corresponding tagged sentence. By using linked list create the node structure for this process. The corresponding node structure as follows,

```
Struct Node
 {
  Char Snt[50];
  Struct Node *Next;
 };
```



Algm_InitialCheck() algorithm is used to compare the input sentences against sentences in supervised Corpus. If the input sentence is present, then the corresponding tagged sentence is retrieved from the corpus.

InputSent←InsertNode(Each words in the input sentence using,InputSent)

Word←InserNode(Words in the each sentence of the Supervised corpus, Word)

Tag←InserNode(Tags in the each sentence of the Supervised corpus,Tag)

If the input sentence is not present in the supervised corpus then the tagging is performed by taking the initial probability [10][11] of the first word in the input sentence. To find the highest probability we create another node structure

```
Struct Node
{
  Char Sent[50];
  Int Count;
  Struct Node *Nxt;
};
```

Sent	Count	Nxt

Then we apply Hidden Markov Model Bi-gram method. We have to find out the POS tag of next word in the sentence by using the previous word and its corresponding tag .Here the comparison done by consider the length of input sentence. By using these data we compared it against the tagged sentences in the corpus for finding the maximum probability of occurrence. Then assign the higher probability tag to the particular word. We are using our own smoothening algorithm to improve the accuracy of the tagged sentence and expand the corpus with newly learned sentences. We are using our own rule based corpus for smoothening purpose. In this corpus we store the combination of tags. These tags are created based on the tag sequence of supervised corpus.

Sample Rule Based corpus is as follows,

Table 2
Rule Based Corpus

Rule Based Corpus
ADV VE PRO SENT #
NO VE #
NO VE VE #
NO PRE AD NO #
NO VE AD NO #
PRO VE PRE DE NO ADV #
PRO VE AD NO CO PRO VE PRO #
IN PRO CO DE AD NO VE PRE NO ADV #
DE VE DE NO #
VE DE DE NO SENT #

Rule based corpus is mainly used to check the accuracy of the tagged sentence. Each combination ends with '#'. After applying HMM Bigram model if we get an incompletely tagged sentence, then the remaining part will be tagged in the rule based corpus. Here we apply our own smoothening algorithm with HMM bigram method up to the end of the sentence.

Algm_Initial_Present() is used to check whether there is a match between the initial word of the inputted sentences and the initial word of sentences in the supervised corpus. Finally, the HMM Bigram method is applied based on the length of the input sentence. If HMM Bigram model fails, the smoothening is performed.

When the initial word is not present in the supervised corpus then **Algm_Initial_NotPresent()** is used to finding the initial probability. Initial probability is calculated using the higher probable initial tag in the rule based corpus. We use another smoothening algorithm for improving the accuracy of the tagged sentence. This algorithm is an enhanced version of HMM Bigram model, which uses Rule based and supervised corpus sentences for smoothening.

InitialProbNodeAdd( ) is used to store the count of corresponding Tag of each word.

Alg_InitialProbNodeAdd(Tag,First)

First is the header node of struct node1

1. Start
2. New =New Node1
3. Temp=New Node1
4. Flag=0
5. if First =NULL Then
   New→Sent=Tag
   New→Nxt=NULL
   First=New
   First→Count=1
   Else
   Temp=First
   Repeat while Temp→Nxt ≠ NULL
   If temp→Sent=Tag Then
   Temp→Count=Temp→Count+1
   Flag=1
   Break
   End If
   End While
   If Flag =0 Then
   If Temp→Sent=Tag Then
   Temp→Count=Temp→Count+1
   Else
   New→Sent=Tag
   New→Nxt=NULL
   Temp→Nxt=New
   Temp→Nxt→Count=1
   End IF
   End If
6. Return First

FindProb() is used to find the Highest probability of the word and its corresponding tag based on the **InitialProbNodeAdd( )** Algorithm.

Algm_FindProb(Word,First,Pnew)

First is the header node of struct node1

1. Start
2. New=New Node
3. Temp=First

4. Largest=Temp→Count
5. Tag=Temp→Sent
6. Repeat While Temp ≠ NULL
   If Largest < Temp→Count
   L=Temp→Count
   Tag=Temp→Sent
   End If
   Temp=Temp→Nxt
   End While
7. Token=Word
8. Concat Token with'\'
9. Concat Token With Tag
10. New→Snt=Token
11. New→Next=NULL
12. Return New

SecProb( ) is used to find the Highest probability [12][13] word and the corresponding tag by HMM Bigram method.

Algm_SecProb(Word,Head,LastNodeHeadResult)

1. Start
2. Structure of Node1 Rnode=NULL;
3. New=New Node
4. Repeat while not end of the rule base corpus
   Repeat while at the end of sentence
   Read each Tag from the corpus into Tag
   Head=InsertNode(Tag,Head)
   End While
   If it's the End of Sentence
   Tag=LastNodeHeadResult→Tag
   Temp=Head
   Succesor=Temp→Next
   Repeat while Temp ≠ NULL and Successor ≠ NULL
   If compare Tag=Temp→Snt
   Rnode=InitialProNodeAdd(succesor→Snt, Rnode)
   End if
   Temp=Temp→Next
   Succesor=Succesor→Next
   EndWhile
   Head=NULL
   End If

End While

5. New =FindProb(Word,Rnode,New)
6. Head=New
7. Return Head

CheckSuperCor( ) is used to check whether the corresponding word is present in the supervised corpus or not.

LengthSuperchk() Algorithm is used to check the presence of a word in the supervised corpus based on the length of input sentence.

Algm_LengthSuperchk(CountOfInp, HeadResult,Length)

1. Start
2. Create an object HdFirst of Structure node1
3. New=Newnode
4. Flag=0
5. Repeat while not end of the supervised corpus

   Repeat while at the end of sentence

   Token=Read each Token

   New=InsertNode(Token,New)

   End While

   If End Of the Sentence Then

   LengthNew=Length of New

   If Length=LengthInp and Flag=0 Then

   Temp ← Last Node of HeadResult

   Lastsnt=Temp→Snt

   Temp=Headinp

   I=0

   Repeat While I< CountOfInp

   I=I+1

   Temp=Temp→Next

   End While

   Snts=Temp→Snt

   Concat Lastsent with " "

   Concat Lastsent with Snts

   Temp=New

   Successor=Temp→Next

   Repeat while Temp ≠ NULL and Successor ≠ NULL

   Token= Successor→Snt

   Word=Word of Token

   Tag=Tag of Token

   cmprData=Temp→Snt

Concat cmprData with " "
Concat cmprData with Word
If Compare lastsent = cmprData Then
HdFirst=InitialProbNodeAdd(Token,HdFirst)
End If
Temp=Temp→Next
Sucsessor=Temp→Next
Clear Token,Word,Tag, cmprData,Lastsent
End while
End If
Else
Flag=1
End If
End If
New=NULL
End While
If HdFirst ≠ NULL Then
New=NewNode
New=FindProb(Snts,HdFirst,New)
Temp=Last node of HeadResult
Temp→Next=New
Flag=0
Else
Flag=1
End If

6. Return Flag

Algorithm InsertNode() is used to create a new node and insert Words and Tags [14][15].

UpdateCorpus( ) is used to update the supervised corpus and rule based corpus with newly learned sentences and corresponding tags.

3. EXPERIMENTAL RESULT

1. how\ADV many\AD brothers\NO do\VE you\PRO have\VE ?\SENT
2. it\PRO was\VE nice\PRO chatting\VE with\PRO you\VE
3. May\NO i\PRO take\VE the\DET passport\NO pleace\VE ?\SENT
4. how\ADV can\VE you\PRO be\VE so\ADV sure\AD ?\SENT
5. the\DE scheme\NO starts\VE out\PRO as\VE planned\PRO

4. CONCLUSION

In this paper we proposed to introduce an enhanced Parts-Of-Speech (POS) tagging approach for tagging English sentences using statistical approach. We use our own smoothening algorithm to increase the accuracy of parts of speech tagging and also using our own Tag set contains 11 Tags. As part of implementation, a

combination of supervised corpus and rule based corpus is used. This algorithm provides better accuracy for the tagged sentence. Finally the corpus update with the newly learned sentence and we get an overall accuracy of 80%.

REFERENCES

[1] Deepika Kumawat and Vinesh jain "POS Tagging Approches: A Comparison" International Journal of computer Applications(0975-8887)volume 118-No.6, May 2015.

[2] Antony P J , Dr. Soman K P " Parts Of Speech Tagging for Indian Languages: A Literature Survey" Volume 34– No.8, November 2011.

[3] P. Brown, V. Della Pietra, P. deSouza, J. Lai, and R. Mercer. Class-based n-gram models of natural language. Computational Linguistics, 1992.

[4] Jean-Pierre Chanod and Pasi Tapanainen. Tagging French – comparing a statistical and constraint-based method. In EACL-95, 1995.

[5] Rajeev R.R. , Jisha P.Jayan , and Elizabeth Sherly " Parts of Speech Tagger for Malayalam" vol.2,NO.2,December 2009,pp.209-213.

[6] Cutting *et al.* 1992. A practical part-of-speechtagger. In Proc. of the Third Conf. on Applied NaturalLanguage Processing. ACL 1992.

[7] Eric Brill. Transformation-Based Learning. PhD thesis, Univ. of Pennsylvania, 1993.

[8] Eric Brill. Some advances in transformation-based part of speech tagging. In Proceedings of AAAI-94, 1994.

[9] Arun Kumar N , Dr. U. Krishnakumar " An Enhanced POS Tagger for English Sentences".

[10] Kuang-hua Chen and Hsin-Hsi Chen. Extracting noun phrases from large-scal texts: A hybrid approach and its automatic evaluation. In Proceedings of ACL,1994.

[11] Kenneth Church. A stochastic parts program and noun phrase parser for unrestricted texts. In Proceedings of the Second Conference on Applied Natural Language Processing, Austin, Texas, 1988.

[12] Doug Cutting, Julian Kupiec, Jan Pedersen, and Penelope Sibun. A practical part-of-speech tagger. In Third Conference on Applied Natural Language Processing (ANLP-92), pages 133–140, 1992.

[13] Evangelos Dermatas and George Kokkinakis. Automatic stochastic tagging of natural language texts. Computational Linguistics, 21(2):137–164, 1995.

[14] Eva Ejerhed and Kenneth Church. Finite state parsing. Seventh Scandinavian Conference of Linguistics,University of Helsinki,Department of General Linguistics,1983.

[15] Helmut Feldweg. Implementation and evaluation of a German HMM for POS disambiguation. In EACL SIGDAT Workshop, 1995.