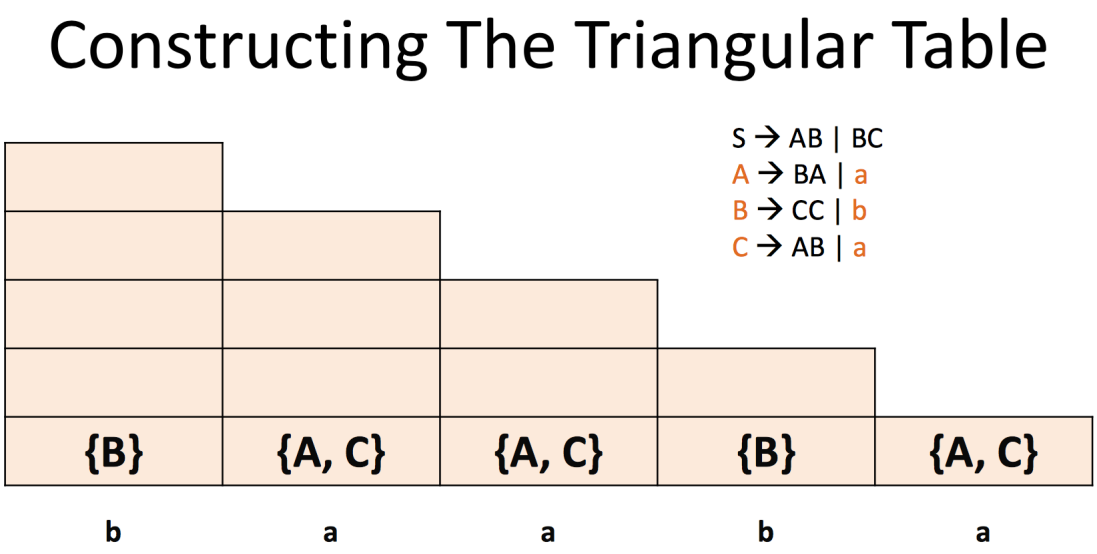
RAKA DALAL

Homework 4

1. In Probabilistic Context Free Grammar, we have a CFG $G = (N, \Sigma, S, R)$ where we have a probability $q(\alpha \rightarrow \beta)$ associated with each $\alpha \rightarrow \beta \quad \in R$. In this case we have multiple possible derivations for a sentence of N words and our objective is to calculate the most probable one given the sentence. However we can calculate the number of possible derivations for a sentence of N words using a simple modification of the CYK algorithm. The modification is the following: Since the rules are probabilistic, we will have multiple options when we are combining the elements in the lower triangular table as we move up the row. Normally what we do is to calculate the sum of the probabilities in which the elements in the rows below can be generated but this time we will also pass up the number of ways in which those elements could have been generated. To give an example consider this picture



where all the cells in the bottom row has been generated already. Now to fill up the cell $X_{1,1}$ we have to see the number of ways $(X_{1,1}, X_{2,2})$ is generated i.e the number of ways (BA, BC) is generated. Now there will exist multiple rules which will lead to BA or BC or both. Hence, for each rule ,

we write the non-terminal and the number of elements in the set (BA, BC) that it can generate. So suppose there are rules like $S \rightarrow BA$ with probability 0.3 and $S \rightarrow BC$ with probability 0.6 and in that case we will write $(S, 2)$ in that cell. These numbers get added up when we move higher up in the rows (so that if we had (B, i) in $X_{1,1}$ and $(A, j), (C, k)$ in $X_{2,2}$ then we would have put $(S, \min(i, j) + \min(i, k))$ in $X_{2,1}$ and the final number that we get in the top cell is the number of derivations of the entire sentence. This algorithm has space complexity $\Theta(N^2)$ and time complexity of $\Theta(N^3)$

2. In this paper "Coarse to fine n-based parsing and MaxEnt Discriminative Ranking" , the main objective of the authors is to develop a scheme for constructing sets of 50 best parses based on a coarse to fine generative parser. This idea of a coarse to fine generative parser was first developed by Charniak in 2010. Then the authors have used a ranker to rank those set of parses for each sentence and the ranker they have used is a MaxEnt Ranker. Since this ranker has to select from a really small group of parses per sentence, it is not necessary to use dynamic programming which allows the features to be essentially arbitrary functions of the parse trees. However the major difficulty in extracting the n best parses instead of simply selecting the best one is that of dynamic programming. Roark(2001) has used beam search without dynamic programming and Collins' (1997) technique of using dynamic programming does not extend to finding the n-best parses. The authors, in order to circumvent these issues , have used a clever technique based on the algorithm of Schwartz and Chow. They have modified the Viterbi algorithm to select the 50 best parses instead of the optimal one. The key insight is that the n'th best parse can only involve n suboptimal parsing decisions and all but one of these must be involved in one of the second through the $n - 1$th best parses. Hence the idea is to iteratively find the second best parse, then the third best parse and so on. This kind of algorithm was first described for the Hidden Markov Models. This results in an increase of space complexity from $\Theta(m^2)$ in finding the optimal parse to $\Theta(nm^2)$ in finding the n best parses where m is the number of states. For bilexicalized sentences , it becomes even worse to $\Theta(nm^3)$ but the coarse to fine parsing nature of the Charniak parser comes to the rescue. On a high level , the idea is to use dynamic programming using CYK algorithm to find the parsing on a very coarse level (using very few states) and finally prune them exhaustively according to a fine grained probabilistic model. The authors ran their algorithm on Section 24 of the Penn WSJ tree-bank and the $f - score$ obtained by them goes up to 0.968 when the value of n is increased upto 50. The authors subsequently have described the features they have used in order to rank the parsers and they have estimated the feature weights by using a regularized loss function and chosen the weight that minimizes the loss function.
   Overall their algorithm is able to use dynamic programming in an efficient manner with a low space complexity. Their algorithm is simple and practical and their experiments also show incredible results indeed and hence I think that this paper is a very important contribution to this field.

3. HW4-Q3.py is the sentence generator and Grammar.txt is the weighted CFG. There are on optional arguments.
   To run the sentence generator:
   python HW4-Q3.py t ROOT-SYMBOL NUM-TO-GEN "Grammar.txt"
   t is either 1 or 0.
   An example run:
   python HW4-Q3.py 0 "Root" 4 "Grammar.txt"

I have used three linguistic phenomena in my Grammar-noun-verb agreement (singular vs. plural) for present tense verbs, "a" vs. "an" and adjective order. Instead of NP, I have used SNP and PNP to denote singular and plural.
2 S –> SNP SVP
1 S –> PNP PVP
For the rest of the grammar, I have maintained this singular and plural division. Next, I had to capture the linguistic phenomenon "a" vs. "an".
2 SNP –> an SVNPish
2 SNP –> a SCNPish
V for vowel and C for consonant.
Next, for adjective order I used this.
1 CAdj –> cquality csize ccolour
1 VAdj –> vquality vsize vcolour
Again C and c for consonant and V and v for vowel.
The lexicon consists of the words "Chris", "eat", "eats", "sandwich", "apple", "Girls", "adopted", "beautiful", "big", "white", "dog", "to","of","apples", "ugly", "enormous", "orange".

The sentence generator produces sentences as:
Chris eats an apple .
Here, eats matches the noun-verb agreement and an apple matches "a" vs. "an" linguistic phenomenon.

Similarly,
Chris adopted a beautiful big white dog .
Here, "a" vs "an" linguistic phenomenon and adjective order is matched.

Another two examples,
dog adopted a sandwich ?
an apple eats girls .
As you can see here, the sentence generator never generates sentences that violates those three linguistic phenomena. But it cannot take into account the meaning of the words, thus the third sentence is produced.

Setting grammar weights is quite a difficult task. I was quite frustrated with the Viterbi parse of certain sentences. I tried to set weights so that it can make meaningful sentences.
3 Root –> S .
1 Root –> S !
1 Root –> S ?
I assigned higher weight to Root –> S . as mostly statements are created.

Next for the sentence generator, I created a list of terminals by following the rule that the terminals can never come on the L.H.S of Grammar. Next for every non terminal, I created a probability distribution from the weights and then sampled a random variable and chose the rule accordingly. I repeated this unless I reached all terminals.

4. In this question I have used the following 2 treebanks for comparison

   - Parallel Universal Dependencies (PUD) treebanks for the Hindi Language
   - Parallel Universal Dependencies (PUD) treebanks for the French Language

   I did 3 comparisons between these languages

   - First I checked how many times in each language each POS tag is appearing (in the entire document) and we showed a joint histogram of this data. I have attached the histogram here for a better visualization. You can observe that Adverbs is used a lot more in French than in Hindi and nouns are used more in Hindi than in French although the usage of Nouns is very high in both the languages. A number of other interesting observations about the 2 languages can also be made from this plot.

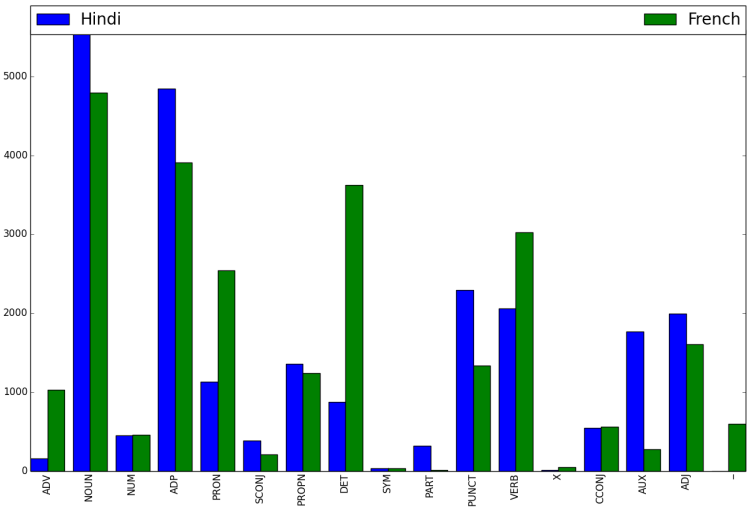


Figure 1: Comaprison of coarse POS tags

   - Secondly I also checked how many times in each language each fine grained POS tag such as RP and NN is used. Again I showed a joint histogram and there are interesting observations there as well
   - Next I tried to analyze how many times the most common words in each language was used in the document. This also shows how rich the language is in terms of vocabulary. So I took the top 20 most common words in each language and showed the histogram of how many times they appeared in each language in an ascending order. You can observe that the most common words in Hindi is used a lot more than its French counterparts which is really interesting.

   Finally I also did a Church and Banks kind of analysis where I tried to see association between words of both French and Hindi through the following empirical formula

$$\text{Word Assoc}(H, F) = \frac{\text{\#Sentences in which both H and F occur}^4}{\text{\#Sentences in which H is used} * \text{\#Sentences in which F is used}}$$

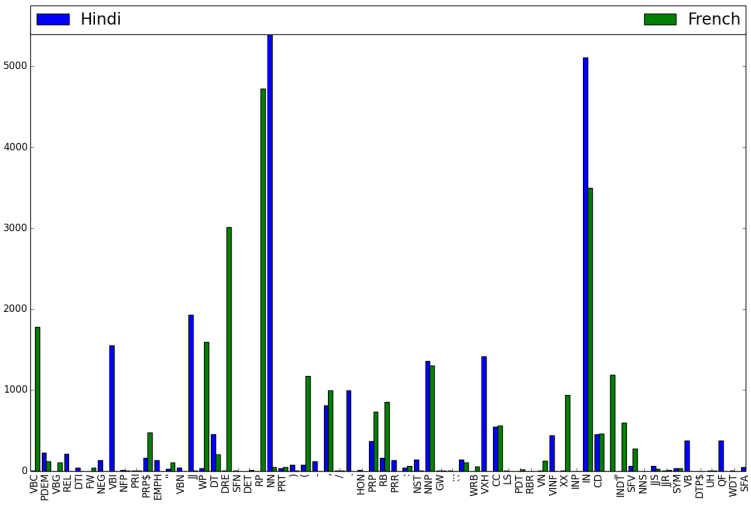


Figure 2: Comaprison of fine grained POS tags

I am taking the power of 4 in the numerator because I wanted to emphasize the common sentences in which they both appear. It turns out that unsurprisingly punctuations in both languages are the most correlated

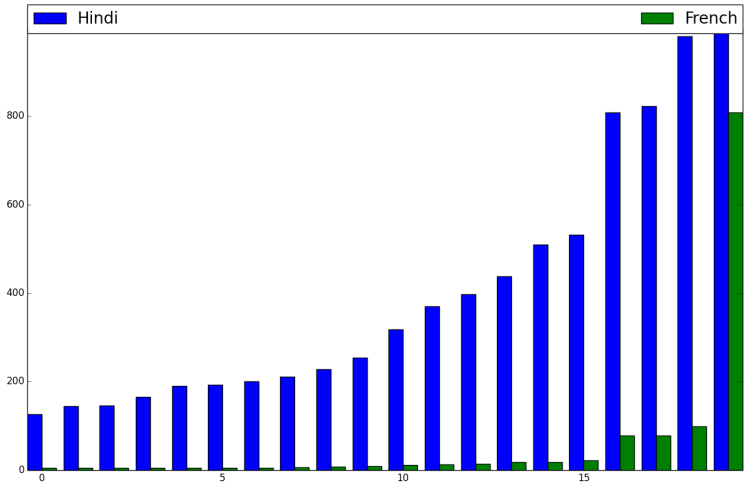


Figure 3: Comparison of most common words