Assignment 1

FIT5217: Natural Language Processing

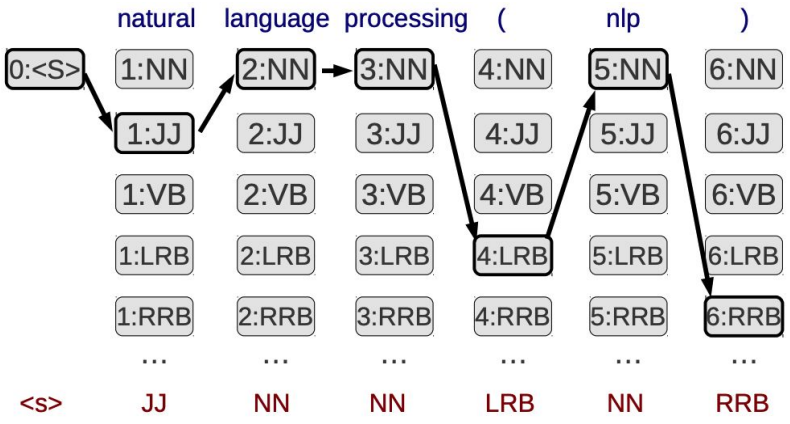
Monash University, TP3, 2020

Objectives

This assignment consists of two parts and covers the first 6 weeks on natural language processing. The total marks of this assessment is 100 which accounts for 20% of the final grade. The objective of this assignment is to increase and check your understanding about sequence labeling and parsing with rules. **Your submissions for the assignment include the code along with a PDF file for Part A and a single PDF file for Part B.**

Part A. POS Tagging with Hidden Markov Model (70 marks)

Our aim is to complete a program for part-of-speech (POS) tagging using hidden Markov Models (HMMs). We will adopt a supervised learning approach in this assignment. You will be given a fully annotated training data of texts along with the POS tags.



The files in the codebase of our HMM-POS tagger on Moodle are as follows:

train-pos.py	To train POS-tagger.
tagger.py	To do POS tagging.
eval.py	To calculate accuracy of the tagger.
train.en	English text for training.
train.pos	POS tag for training.
test.en	Input text to be tagged.
test.pos	Correct POS tag.

Our HMM consists of the following parameters:

- The transition probability $B(y_t \mid y_{t-1})$: The probability of the POS tag for the current position y_t given the POS tag of the previous position y_{t-1} .

- The emission probability $A(x_t|y_t)$: The probability of emitting the word x_t given the POS tag y_t .

As we saw in the lecture slides, the probability of sequence of POS tags $(y_1, y_2, y_3 \ldots y_n)$ for the sentence $(x_1, x_2, x_3, \ldots, x_n)$ according to our HMM is:

$$p(y_1, \ldots, y_n) = \prod_{t=1}^{n+1} B(y_t|y_{t-1})\, A(x_t|y_t)$$

We can set y_0 =<SOS> and x_0 =<SOS>. Using this, we can compute the probability of tags appearing first in a sentence. It is also convenient to use y_{n+1} =<EOS> and x_{n+1} =<EOS> to model how likely tags appear last in a sentence.

With the above background, please work on the following problems:

1. Assume that we are given a training set of sentences and their POS tags $\{(x_i, y_i)\}_{i=1}^{N}$ where $x_{i,j}$ is the j-th word of the i-th sentence, and $y_{i,j}$ is its corresponding POS tag. On the PDF report, write down the maximum-likelihood estimation rule for the HMM parameters, i.e. “A” and “B”. (5 marks)

2. Please complete the maximum likelihood estimation algorithm of POS tagger for training the POS tag in **train-pos.py**. Looking at the equation above, MLE needs to compute two parts: the transition B(.) and the emission A(.). Please collect statistics needed to compute B(.) and A(.) from train-pos.py and output the probability table into dictionaries. (20 marks)

   Hint: To compute conditional probabilities you need to compute the occurrence of a token with its context.

3. Given a trained HMM, write the Viterbi algorithm in **tagger.py** for finding the highest probability sequence of the POS tags for a given test sentence $[x_1, x_2, \ldots, x_n]$. Write the pseudocode of the viterbi algorithm in your PDF report. (25 marks)

   Hint: You need to be careful for input that is not seen in the training data. Please refer to the lecture slides for this.

4. Complete the eval.py to calculate the evaluation metrics. This program will take your prediction (test.hyp) and the ground truth (test.pos) and calculate the accuracy of your tagger and report it in the PDF! (5 marks)

5. We have discussed in the lectures of Week 3, how the unknown words can be handled. Please explain the technique which you used in your code to handle unknown words. Do you need to handle unknown tags? Why or why not? Please explain in the PDF report. (5 marks)

6. In the preprocessing phase of the provided codebase, we have lowercase the data (using the lower() function in python). Why do we perform lowercasing when training / inferencing? Do we always need to do that? Please explain in the PDF report. (5 marks)

7. This assignment uses the first order markov model which models transition probabilities only from the previous tag. While effective, this approach is sub-optimal. Please describe your idea to improve the accuracy of the model in the PDF report. (5 marks)

Part B: Parsing (30 marks)

B1. Chart Parsing (20 marks)

Consider the following grammar :

a.	**S -> NP VP**	**// declarative sentence**
b.	**S -> VP**	**// command**
c.	**NP -> Det Nbar**	**// simple NP with specifier**
d.	**NP -> Nbar**	**// unspecified NP**
e.	**NP -> NP PP**	**// NP with adjunct**
f.	**Nbar -> Noun**	**// simple noun**
g.	**PP -> Prep NP**	**// standard PPs**
h.	**VP -> VP PP**	**// adjuncts**
i.	**VP -> Verb NP**	**// verb with complement (object)**
j.	**VP -> Verb**	**// verb without complement**
k.	**Det -> an | a**	
l.	**Noun -> I | elephants | pajamas**	
m.	**Prep -> in | on**	
n.	**Verb -> shot | saw**	

Warning: You'll note that this grammar is NOT binarized. This means that you'll need to be able to handle unary rules.

Using the above grammar, answer the questions in the following page:

(i) Construct the complete chart for parsing **"$_0$ I $_1$ shot $_2$ elephants $_3$ in $_4$ pajamas $_5$"** using the grammar. (16 marks)

Hint. We have done parts of the very first step, and you should do the rest.

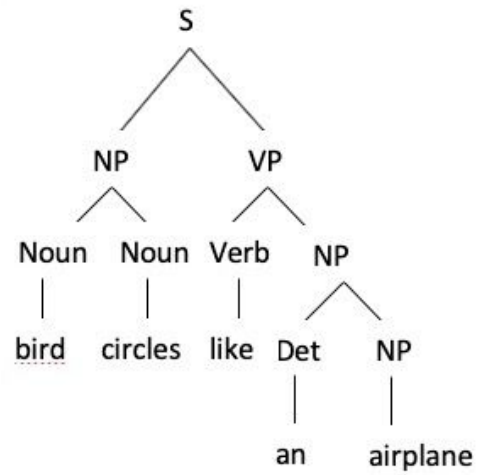
- **Phrases of length 1:**
  - (0,1): Noun[l], Nbar[f], NP[d], ???
  - (1,2): Verb[n], Vp[j], ???
  - (2,3): ???
  - (3,4): ???
  - (4,5): ???
- **Phrases of length 2:**
  - (0,2): S[a,1], ???
  - (1,3): ???
  - ???
- **Phrases of length 3:**
  - ???
- **Phrases of length 4:**
  - ???
- **Phrases of length 5:**
  - ???

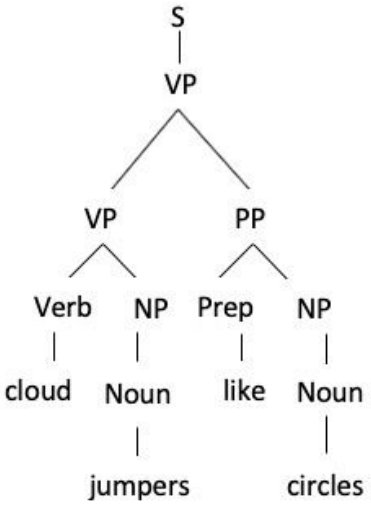
You do the rest, as well as the "???"s above, which may contain stuff and also may not. The notation "X[a,3]" means that you derived something of type "X" using rule "a" and the split point was 3. If you can derive the same thing in multiple ways, write it as X[a, 3]^4 , meaning that you could derive it four ways. (In practice, you would store backpointers, but we'll not do that here.)

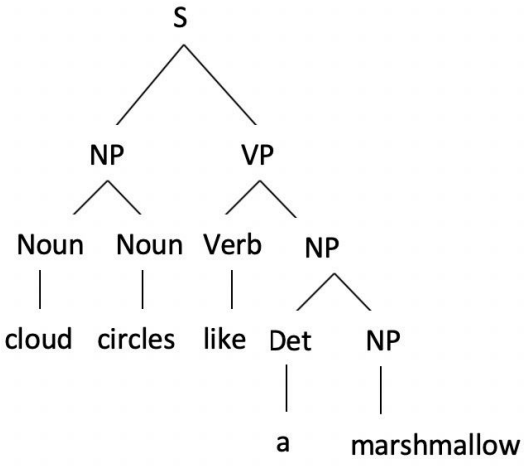
(ii) How many full-sentence parses do you find for this sentence? Explain the unambiguous meaning of the parses you find. (4 marks)

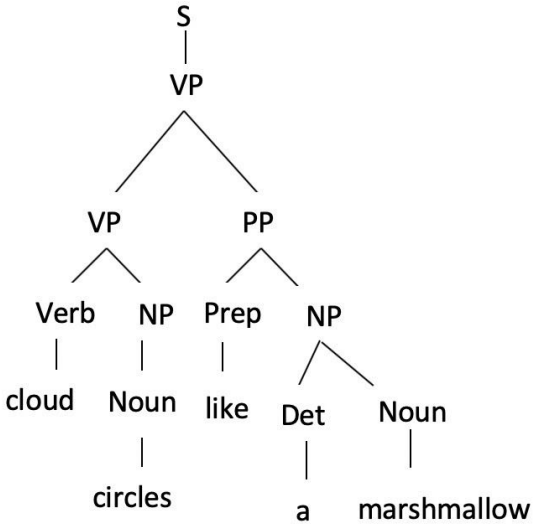
B2. Treebanking (10 marks)

Extract all the context free grammar rules from the following treebank as well as compute the probabilities for those rules according to maximum likelihood. You do not need to smooth.









Submission & Due Date:

The files that you need to submit are:

- For Part A, please submit the whole python script and adhere to the execution guide that is mentioned in the top of the script. Please submit “train-pos.py”, “tagger.py”, and “eval.py”. Additionally Please also submit your “test.hyp” generated by the “tagger.py”. Also submit “PartA_solution.pdf”.
- For Part B, you should submit the “PartB_solution.pdf” file. You must add enough comments to your solution to make it readable and understandable by the marker.
- Submit one single zip file containing your code and PDF files described above. Do not need to submit the training and the testing data. Please name your zip file**: STUDNETID_assignment.zip**

Assessment Criteria:

The following outlines the criteria which you will be assessed against:

- Ability to understand the fundamentals of POS tagging and parsing.
- Working code: The code executes without errors and produces correct results.
- Understand how to conduct POS and parsing for text.

Penalties:

- Late submission (-20% of awarded marks for between 1-3 days late pro-rata, -50% for between 4-6 days late pro-rata. 0 marks awards after 6 days)
- Submitted file (zip file name, py script, .ipynb file) is not properly named (-5%)
- Results and predictions are not reproducible. (-25%)

Acknowledgment:

We thank Graham Neubig from Carnegie Mellon University and Hal Daume from University of Maryland or their slides and materials for the content of this assignment.