CS6320, Spring 2019
Dr. Mithun Balakrishna
Homework 2
Due Sunday, February 24th, 2019 11:59pm

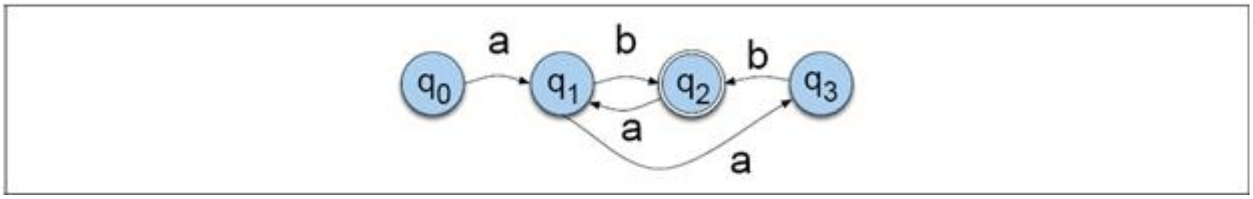
A. Submission Instructions:

- Submit your solutions via eLearning.
- Please submit a single zip file with the following files:
  - For programming questions:
    - Source code file(s) in C/C++, Java, or Python. For using any other programming language, please get prior approval from the TA.
    - A ReadMe file with instructions on how to compile/run the code.
  - For all other questions, a PDF/Doc/PS/Image file with the solutions.
- Late Submission Penalty:
  - up to 2 hours late — 10% deduction
  - 2 - 4 hours late — 20% deduction
  - 4 - 12 hours late — 35% deduction
  - 12 - 24 hours late — 50% deduction
  - 24 - 48 hours late — 75% deduction
  - more than 48 hours late — 100% deduction (zero credit)

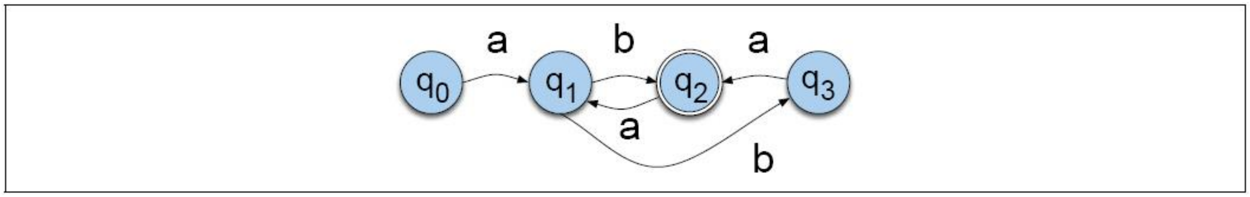
B. Problems:

1. NFSA to Regular Expression (20 points)

a. (**10 points**) Write a regular expression for the language accepted by the FSA:



b. (**10 points**) Write a regular expression for the language accepted by the NFSA:



2. Bigram Probabilities (40 points):

Write a computer program to compute the bigram model (counts and probabilities) on the given corpus (*HW2_F17_NLP6320-NLPCorpusTreebank2Parts-CorpusA.txt* provided as Addendum to this homework on eLearning) under the following three (3) scenarios:

i. No Smoothing
ii. Add-one Smoothing
iii. Good-Turing Discounting based Smoothing

Note:

1. Use the " . " string sequence in the corpus to break it into sentences.

2. Each sentence should be tokenized into words and the bigrams computed ONLY within a sentence.

3. Please use whitespace (i.e. space, tab, and newline) to tokenize a sentence into words/tokens that are required for the bigram model.

4. Do NOT perform any type of word/token normalization (i.e. stem, lemmatize, lowercase, etc.).

5. Creation and matching of bigrams should be exact and case-sensitive.

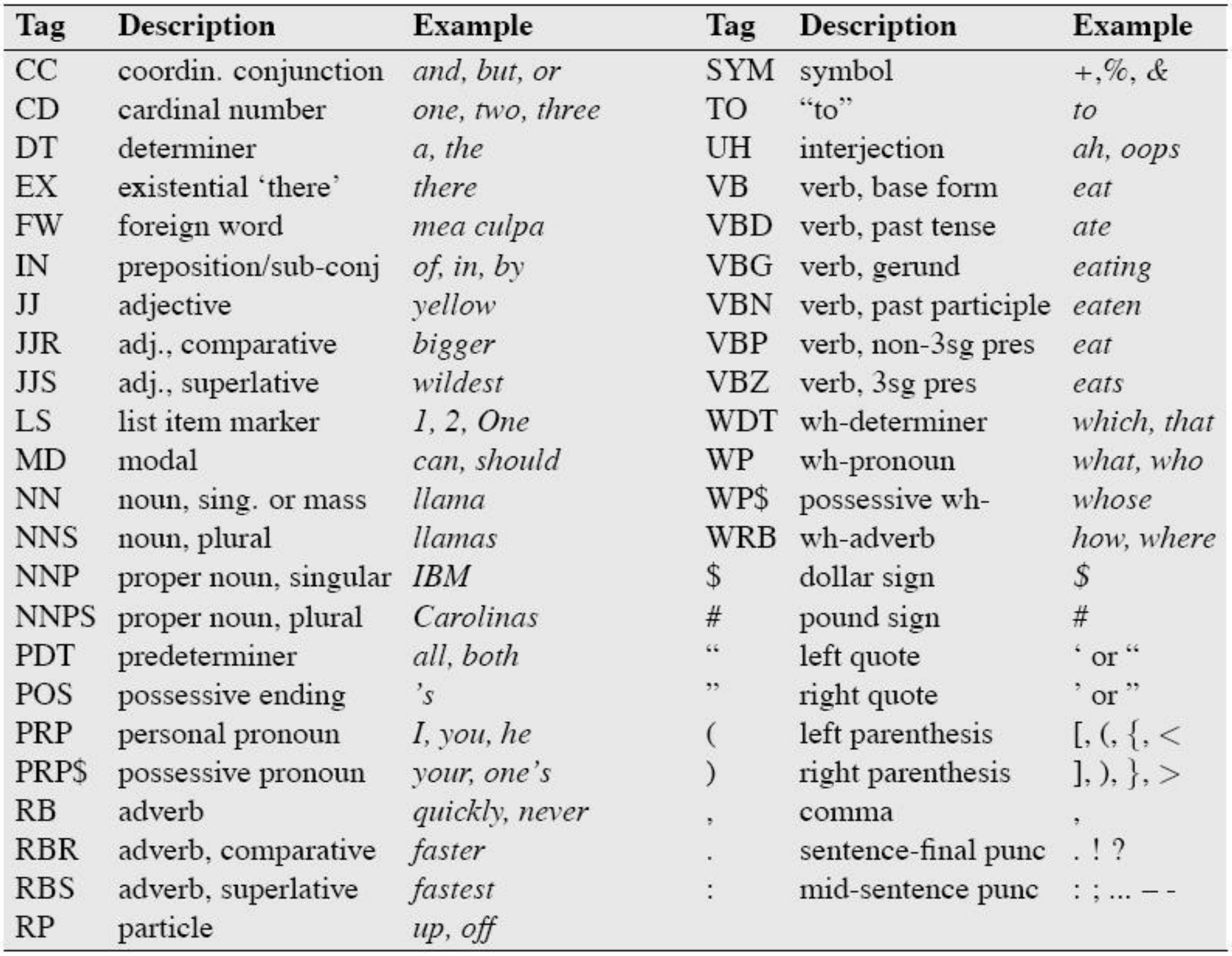
Input Sentence: *The Fed chairman warned that the board 's decision is bad*

Given the bigram model (for each of the three (3) scenarios) computed by your computer program, **hand** compute the total probability for the above input sentence. Please provide all the required computation details.

Note: Do NOT include the unigram probability P("The") in the total probability computation for the above input sentence.

3. Transformation Based POS Tagging (40 points)

For this question, you have been given a POS-tagged training file, *HW2_F17_NLP6320_POSTaggedTrainingSet.txt* (provided as Addendum to this homework on eLearning), that has been tagged with POS tags from the Penn Treebank POS tagset (Figure 1).



Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	*and, but, or*	SYM	symbol	+,%, &
CD	cardinal number	*one, two, three*	TO	“to”	*to*
DT	determiner	*a, the*	UH	interjection	*ah, oops*
EX	existential ‘there’	*there*	VB	verb, base form	*eat*
FW	foreign word	*mea culpa*	VBD	verb, past tense	*ate*
IN	preposition/sub-conj	*of, in, by*	VBG	verb, gerund	*eating*
JJ	adjective	*yellow*	VBN	verb, past participle	*eaten*
JJR	adj., comparative	*bigger*	VBP	verb, non-3sg pres	*eat*
JJS	adj., superlative	*wildest*	VBZ	verb, 3sg pres	*eats*
LS	list item marker	*1, 2, One*	WDT	wh-determiner	*which, that*
MD	modal	*can, should*	WP	wh-pronoun	*what, who*
NN	noun, sing. or mass	*llama*	WP$	possessive wh-	*whose*
NNS	noun, plural	*llamas*	WRB	wh-adverb	*how, where*
NNP	proper noun, singular	*IBM*	$	dollar sign	*$*
NNPS	proper noun, plural	*Carolinas*	#	pound sign	#
PDT	predeterminer	*all, both*	“	left quote	‘ or “
POS	possessive ending	*’s*	”	right quote	’ or ”
PRP	personal pronoun	*I, you, he*	(	left parenthesis	[, (, {, <
PRP$	possessive pronoun	*your, one’s*	)	right parenthesis	],), }, >
RB	adverb	*quickly, never*	,	comma	,
RBR	adverb, comparative	*faster*	.	sentence-final punc	. ! ?
RBS	adverb, superlative	*fastest*	:	mid-sentence punc	: ; ... – -
RP	particle	*up, off*			

Figure 1. Penn Treebank POS tagset

Use the POS tagged file to perform:

a. Transformation-based POS Tagging: Implement Brill’s transformation-based POS tagging algorithm using ONLY the previous word’s tag to extract the **best** transformation rule to:

   i. Transform “NN” to “JJ”

   ii. Transform “NN” to “VB”

Using the learnt rules, fill out the missing POS tags (for the words “*standard*” and “*work*”) in the following sentence:

*The*_DT *standard*_?? *Turbo*_NN *engine*_NN *is*_VBZ *hard*_JJ *to*_TO *work*_??

b. Naïve Bayesian Classification (Bigram) based POS Tagging:

$$\hat{t}_1^n = \operatorname*{argmax}_{t_1^n} P(t_1^n|w_1^n) \approx \operatorname*{argmax}_{t_1^n} \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1})$$

Using the given corpus, write a computer program to compute the bigram models (counts and probabilities) required by the above Naïve Bayesian Classification formula.

Using the created bigram models, **hand** compute the missing POS tags (for the words "*standard*" and "*work*") in the following sentence:

*The*_DT *standard*_?? *Turbo*_NN *engine*_NN *is*_VBZ *hard*_JJ *to*_TO *work*_??