NIKITA VISPUTE
NET ID: NXV170005
UTD email: nxv170005@utdallas.edu
CS 6320.002

Homework 1

Problem 1: Regular Expressions

1. Language 1:

The set of all strings with two consecutive repeated words (e.g., "Humbert Humbert" and "the the" but not "the bug" or "the big bug").

Answer: **([a-zA-Z] +) \s + \1**

- The characters inside [] indicate a disjunction for the pattern to be matched and the '-' gives the range of characters required to be matched.
- The Kleene + operator indicates one or more occurrence of the previous character.
- The '()' specify that the given operation is only for the pattern within the braces.
- '\s' expression repression a whitespace character along with the Kleene + operator indicating one or more occurrence of whitespace characters.
- '\1' expression indicates the repetition of the immediate previous pattern.

Thus, in all the regular expression identifies text that lies between the range of lower case 'a' to lower case 'z' or between the range of upper case 'A' to upper case 'Z'. This pattern can be repeated since there is the Kleene + operator at the end and within the braces. Next the pattern can be followed by one or more white spaces or tab since there is a Kleene + operator after the white space character. Lastly, \1 represents the repetition of the immediate previous pattern.

2. Language 2:

All strings that start at the beginning of the line with an integer and that end at the end of the line with a word.

Answer: **^\d+\b.*\b [a-zA-Z] + $**

- The caret ^ operator at the beginning of the expression searches for the immediate next character only at the start of the line.
- '\d' expression indicates any digit.
- '\b' expression searches only for the character that is given within the \b__\b expression.
- The dot '.' represents any character.
- The Kleene * operator specifies 0 or more occurrences of the previous character.
- The characters inside [] indicate a disjunction for the pattern to be matched and the '-' gives the range of characters required to be matched.
- The Kleene + operator indicates one or more occurrence of the previous character.
- The dollar operator '$' searches for the immediately preceding character only at the end of the line.

Thus, the regular expression identifies patterns in which there is a digit at the beginning of the line with possibly one or more occurrences of digits followed by 0 or more occurrences of any other character/s and then followed by 1 or more occurrences of a string of text at the end of the line.

3. Language 3:
All strings that have both the word “grotto” and the word “raven” in them (but not, e.g., words like grottos that merely contain the word grotto).
Answer: **\b grotto \b .* \b raven \b | \b raven \b .* \b grotto \b**

- ‘\b’ expression searches only for the character that is given within the \b__\b expression.
- The dot ’.’ represents any character.
- The Kleene * operator specifies 0 or more occurrences of the previous character.
- The ‘|’ operator means disjunction: OR

Thus, the regular expression identifies patterns in which the sentence has the words grotto and then raven with 0 or more occurrences of any other character in between or vice-versa.

Problem 2: N-Grams

Corpus: Downloaded from the webpage
https://personal.utdallas.edu/~sanda/courses/NLP/cs6320.html and saved as ‘Corpus.txt’ locally in the same directory as the program.
S1: Sales of the company to return to normalcy.
S2: The new products and services contributed to increase revenue.
The corpus filename, S1, S2 are given as arguments for the program HW1_NGrams.py

Output for Part 1-4 of Problem 2 program is stored in “output-NGRAMS.txt”
To execute program: HW1_NGrams.py, README-NGRAMS.txt

Which of the two sentences S1, S2 are more probable under

a) Use the bigram model without smoothing:
Sentence 1 is more likely than Sentence 2 since its total probability is higher than the total probability of Sentence 2.
Probability of S1: 2.8405433054215833e-11
Probability of S2: 2.509765608337218e-12

b) Use the bigram model with add-one (Laplace) smoothing:
Sentence 1 is more likely than Sentence 2 since its total probability is higher than the total probability of Sentence 2.
Probability of S1: 1.2995786381836214e-20
Probability of S2: 2.63112641349879e-26

Since probabilities are equal to or lie between the range of 0 to 1. The total probability for a sentence becomes smaller as the bigram probabilities are multiplied to each other. Hence, for comparison logarithm of probabilities is easier to use.

Thus, using log probabilities, since log probability of Sentence 1 is lesser than log probability Sentence 2, Sentence 1 is more likely. Similarly, for the Laplace-smoothed probabilities as well. Also notice that the Laplace-smoothed total probabilities for both sentences is smaller than the probabilities without smoothing for both sentences.

Sentence No	Without smoothing total log probability:	Laplace-smoothed total log probability:
S1	0.0022055211635133745	0.000330610928974661
S2	0.14374531669733176	0.0003699374584481377

Problem 3: Vector Semantics

Corpus: Downloaded from the webpage https://personal.utdallas.edu/~sanda/courses/NLP/cs6320.html and saved as 'Corpus.txt' locally in the same directory as the program.
The glove file "glove.6B.zip" was downloaded from https://nlp.stanford.edu/projects/glove/ and after unzipping it the "glove.6B.50D.txt" was used. The "glove.6B.50D.txt" corpus should also be saved in the same directory as the program. (required for part 4 of the Problem)
The corpus filename is given as argument for the program HW1_VectorSemantics.py

Output for Part 1-4 of Problem 3 program is stored in "output-VECTORSEMANTICS.txt"
To execute program: HW1_VectorSemantics.py, README-VECTORSEMANTICS.txt

1. Compute the PPMI and populate the term-context matrix for:

Term Context Matrix for PPMI with no smoothing

	said	of	board
chairman	0.207	0.	0.
company	0.	0.	1.238

a. The word "chairman" for the *context-word* "said":	0.207
b. The word "chairman" in the *context-word* "of":	0
c. The word "company" in the *context-word* "board":	1.238
d. The word "company" in the *context-word* "said":	0

2. Use add-2 smoothing for the same words and contexts:

Term Context Matrix for PPMI with add-2 smoothing

	said	of	board
chairman	0.218	0.	0.
company	0.	0.804	1.143

a. The word "chairman" for the *context-word* "said":	0.218
b. The word "chairman" in the *context-word* "of":	0
c. The word "company" in the *context-word* "board":	1.143
d. The word "company" in the *context-word* "said":	0

3. Find which words are more similar among:
   [chairman, company], [company, sales] or [company, economy] when considering only the contexts provided by the *context-words* "said", "of", and "board"?

Count Matrix for [chairman, company, sales and economy] in context with ['said', 'of', 'board']

	said	of	board
chairman	125	0	16
company	24	0	29
sales	3	0	0
economy	2	0	0

[chairman, company] similarity : 0.73
[company, sales] similarity : 0.64
[sales, economy] similarity : 1.0
Sales, economy are found to be most similar among others in context of said, of, board.

According to the count matrix and cosine similarity computed,

- The word chairman and company have a lot of occurrences in context of the words 'said' and 'board' yet the similarity computed is 0.73. This indicates that although the words are fairly similar, since the occurrences of chairman are a lot more than company in the context of the words 'said' and 'board', it could be that the word company is associated with word chairman many times but the word chairman may be associated with other words/meanings as well in context of the given context words according to the corpus.
- The word company has a lot of occurrences in context of the words 'said' and 'board' but the word sales does not, yet the similarity computed is 0.64. This indicates that the words are averagely similar, since the occurrences of company are a lot more than sales in the context of the words 'said' and 'board', it could be that the word sales is associated with word company many times but the word company has other meanings/associations as well according to the corpus and context-words given.
- The word sales and economy are most similar as computed from the similarity which is 1.0. This indicates that the occurrences of both words with the given context words in the corpus has very similar meanings and associations. It can also be noticed that both words have 0 count in context with board and of but very close count of 3,2 respectively in context of said from the corpus.

4. Using the pre-trained Glove embeddings to determine which words are more similar among:
   [chairman, company], [company, sales] or [company, economy]?

[chairman, company] similarity : 0.57379776
[company, sales] similarity : 0. 7634718
[sales, economy] similarity : 0.6253733

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus. For this part of the Problem, I have used the Wikipedia 2014 pre-trained word vectors. The similarity of the words [chairman, company], [company, sales] or [company, economy] is computed based on this data and by using a pre-existing model from the genism library to find out the similarity.
According to the statistics given for similarity the words company, sales are most similar. This could be because the Wikipedia corpus pre-trained vectors have a lot of instances where sales and company are used in close correlation.
Since the Wikipedia Glove trained vectors are based on a very large corpus and because the GloVe algorithm is widely used, these results may be more normalized/ accurate and The GloVe model is trained on the non-zero entries of a global word-word co-occurrence matrix, which tabulates how frequently words co-occur with one another in a given corpus as opposed to the small corpus and cosine similarity function considered for the calculation in part 3 which is quite small-scale.

Problem 4: Part-Of-Speech Tagging

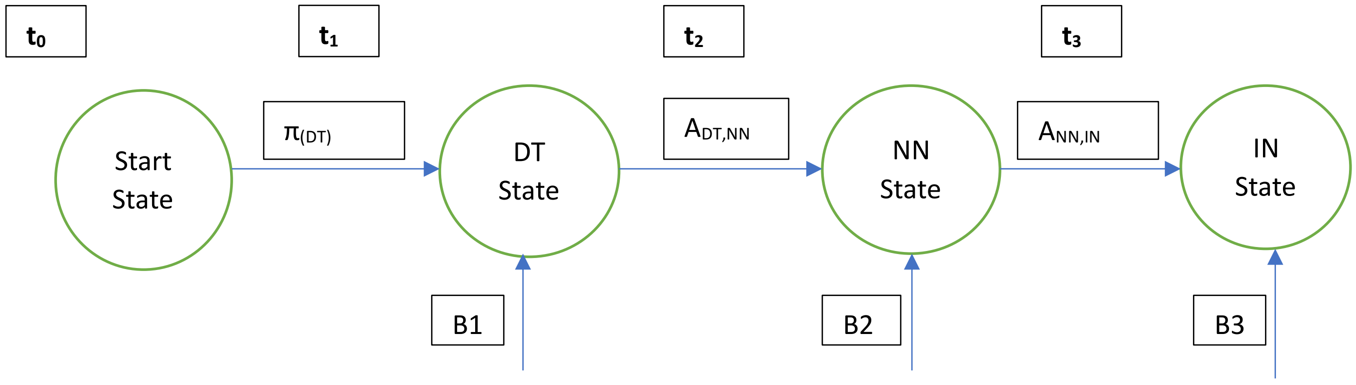
The S1, S2 are given as argument for the program HW1_POSTagging.py
The Stanford POS Tagger full English version is downloaded from
https://nlp.stanford.edu/static/software/tagger.shtml
and the rest of the instructions for its use are in the README.txt file.

Output for Part 1-4 of programming Problem 4 is stored in "POSTAGGING-output.txt"
To execute Problem 4 program: HW1_POSTagging.py, README-POSTAGGING.txt

Use the Viterbi algorithm to assign POS tags to the following two sentences:
S1: The chairman of the board is completely bold.
S2: A chair was found in the middle of the road.

1. Create the Hidden Markov Model (HMM) and show (a) the transition probabilities and (b) observation likelihoods in each state that will be reached by sentences S1 and S2 after 3 time-steps. Present only the transition and observation likelihoods in the states reached after three steps

Sentence 1: ***The chairman of the board is completely bold.***
State Diagram for S1 in 3 time-steps:



In 3 time-steps the first 3 words of the sentence are screened and tagged.
Thus, transition and emission probabilities are recorded for these 3 states only.

Transition Matrix for S1 in 3 time steps:

	DT	NN	VB	VBZ	VBN	JJ	RB	IN
the	0.38	0	0	0	0	0	0	0
chairman	0	0.22	0	0	0	0	0	0
of	0	0	0	0	0	0	0	0.55

Observation Likelihood Matrix for S1 in 3 time steps:

	DT	NN	VB	VBZ	VBN	JJ	RB	IN
the	1	0	0	0	0	0	0	0
chairman	0	1	0	0	0	0	0	0
of	0	0	0	0	0	0	0	1

P(The chairman of) = P(π(DT) * $A_{DT,NN}$ * $A_{NN,IN}$)

= [{(0.38*1) * (0.58*1) }* (0.25*1)]

= 0.22 * 0.25

= 0.55

(1 is from the emission probabilities)

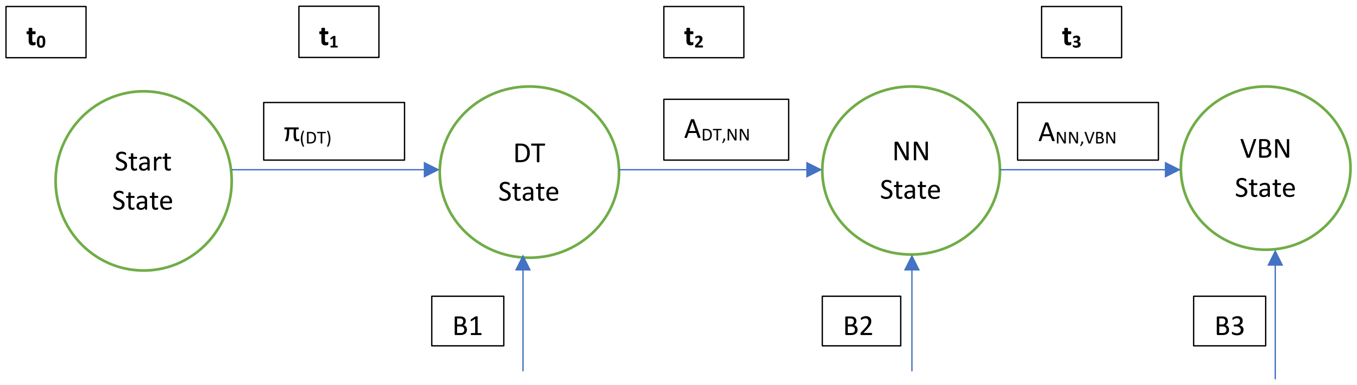
B1 Vector = P(the | DT,), P(chairman | DT), P(of | DT) = 1 , 0 , 0

B2 Vector = P(the | NN,), P(chairman | NN), P(of | NN) = 0, 1, 0

B3 Vector = P(the | IN,), P(chairman | IN), P(of | IN) = 0, 0, 1

Sentence 2: *A chair was found in the middle of the road*.

State Diagram for S1 in 3 time-steps:



In 3 time-steps the first 3 words of the sentence are screened and tagged.
Thus, transition and emission probabilities are recorded for these 3 states only.

Transition Matrix for S2 in 3 time steps:

	DT	NN	VB	VBZ	VBN	JJ	RB	IN
the	0.38	0	0	0	0	0	0	0
chairman	0	0.152	0	0	0	0	0	0
of	0	0	0	0	0.049	0	0	0

Observation Likelihood Matrix for S1 in 3 time steps:

	DT	NN	VB	VBZ	VBN	JJ	RB	IN
the	1	0	0	0	0	0	0	0
chairman	0	0.69	0.31	0	0	0	0	0
of	0	0	0	0	1	0	0	0

P(A chair was) = P(π(DT) * $A_{DT,NN}$ * $A_{NN,VBN}$)

= [{(0.38 *1) * (0.58* 0.69) } * (0.25*1)]

= 0.152 * 0.25

= 0.049

(1, 0.69 is from the emission probabilities)

B1 Vector = P(A | DT,), P(chair | DT), P(was | DT) = 1, 0, 0

B2 Vector = P(A | NN,), P(chair | NN), P(was | NN) = 0, 0.69, 0

B3 Vector = P(the | IN,), P(chair | IN), P(was | IN) = 0, 0, 1

3. What is the probability of assigning the tag sequence for each of the sentences?

Part-of-Speech Tagging for the sentence1: *The chairman of the board is completely bold*
['DT', 'NN', 'IN', 'DT', 'NN', 'VBZ', 'RB', 'JJ']

Probability of assigning tag: 2.90e-05

Part-of-Speech Tagging for the sentence2: *A chair was found in the middle of the road*
['DT', 'NN', 'VBN', 'VBN', 'IN', 'DT', 'NN', 'IN', 'DT', 'NN']

Probability of assigning tag: 1.91e-05

4. Execute the Stanford POS-tagger on both sentences. Which POS tags were assigned by the Stanford POS-tagger more accurately? Explain.

Using Stanford POS Tagging, Sentence 1 is tagged as:
[('The', 'DT'), ('chairman', 'NN'), ('of', 'IN'), ('the', 'DT'), ('board', 'NN'), ('is', 'VBZ'), ('completely', 'RB'), ('bold', 'JJ')]

Using Stanford POS Tagging, Sentence 2 is tagged as:
[('A', 'DT'), ('chair', 'NN'), ('was', 'VBD'), ('found', 'VBN'), ('in', 'IN'), ('the', 'DT'), ('middle', 'NN'), ('of', 'IN'), ('the', 'DT'), ('road', 'NN')]

According to the Stanford POS Tagging and the one done by the Viterbi algorithm,
Sentence 1 has been correctly tagged with the part-of speech.
Sentence 2 has been more accurately tagged by the Stanford POS tagger than the Viterbi algorithm because “was” is a past tense of verb ‘is’ and hence it should be tagged as VBD instead of VBN which is for past participle.

This is because the Stanford POS Tagger model has been trained extensively and hence the model is able to differentiate the POS tags efficiently unlike the small-scale Viterbi algorithm implemented in the program. Also, the Viterbi algorithm is performing the POS Tagging based on the Matrices A, B provided in the Problem. In this, “VBD” tag does not exist hence the Viterbi algorithm is not able to assign “VBD” tag to “was”.

Extra-Credit Problem 2: Neural Language Model

Python Notebook Link for the Neural Language Model

https://colab.research.google.com/drive/1ybtnvnrV0CJ4YybIJDTRvj_QwjOA9EOd

Also attached a python .py file of the same program in the zipped folder named as "HW1_NeuralLanguageModel.py"