Workshop 2

Corpus Processing using NLTK (5%)

In this workshop, you will gain basic skills to perform corpus processing tasks in Python.

Question 1 (10 marks)

Define a conditional frequency distribution over the Names Corpus that allows you to see which initial letters are more frequent for males versus females.

Question 2 (40 marks)

Consider the attached file as your corpus **`warlordofmars.txt`**. Write a program to:

a) Write a function that finds the 50 most frequently occurring words of a text that are not stopwords.
b) Write a program to print the 50 most frequent bigrams (pairs of adjacent words) of a text, omitting bigrams that contain stopwords.
c) Store the n most likely words in a list words, then randomly choose a word from the list using **random.choice()**. (You will need to import random first.)
d) Train a model on this corpus and get it to generate random text. You may have to experiment with different start words. How intelligible is the text? Discuss the strengths and weaknesses of this method of generating random text.

Question 3 (50 marks)

Write a program called **`noun_similarity.py`** that uses the *predefined path-based similarity measure* to score the similarity of each of the following pairs of words:
car-automobile, gem-jewel, journey-voyage, boy-lad, coast-shore, asylum-madhouse, magician-wizard, midday-noon, furnace- stove, food-fruit, bird-cock, bird-crane, tool-implement, brother-monk, lad- brother, crane-implement, journey-car, monk-oracle, cemetery-woodland, food-rooster, coast-hill, forest-graveyard, shore-woodland, monk-slave, coast-forest, lad-wizard, chord-smile, glass-magician, rooster-voyage, noon-string.

In noun_similarity.py implement the function **`get_similarity_scores(pairs)`** so that it ranks the pairs in order of decreasing similarity. Hint: the similarity of a pair should be represented by the similarity of the most similar pair of synsets they have.

Submission

Submit your code and output on Blackboard.