Winter 2019 (Jan7-Apr8)
Faculty of Computer Science

CSCI 4152/6509 — Natural Language Processing

Assignment 3

Due: *Friday, Mar 22, 2019 by midnight*
Worth: 132 marks $(12 + 15 + 20 + 25 + 35 + 25)$
Instructor: Vlado Keselj, CS bldg 432, 902.494.2893, vlado@dnlp.ca

Assignment Instructions:

All answers must be submitted through the SVN by the duedate.

The Lab questions must be submitted in the appropriate directories as specified in the labs (and questions).

The answer files for the other questions, should be submitted in the SVN directory *CSID*/`a3`, similarly to the previous assignment.

All files must be plain-text files, unless specified differently by the question.

1) (12 marks, files in *CSID*/`lab5`) Complete the Lab 5 as instructed, and submit files in your course SVN repository, in the directory *CSID*/`lab5`, where *CSID* is your CSID. In particular, you will need to properly:

a) (3 marks) Submit the file ‘`list_merge.py`’ as instructed in Step 2.

b) (3 marks) Submit the file ‘`stop_word_removal.py`’ as instructed in Step 4.

c) (3 marks) Submit the file ‘`explore_corpus.py`’ as instructed in Step 7.

d) (3 marks) Submit the file ‘`movie_rev_classifier.py`’ as instructed in Step 8.

2) (15 marks, files via SVN in the directory *CSID*/`lab6`) Complete the Lab 6 as instructed. In particular, you will need to properly:

a) (3 marks) Submit the file ‘`hmm_tagger.py`’ as instructed in Step 2.

b) (3 marks) Submit the file ‘`crf_tagger.py`’ as instructed in Step 3.

c) (3 marks) Submit the file ‘`brill_demo.py`’ as instructed in Step 4.

d) (3 marks) Submit the file ‘`ne_chunker_exercise.py`’ as instructed in Step 5.

e) (3 marks) Submit the file ‘`first_notebook.ipynb`’ as instructed in Step 6.

3) (20 marks, files in *CSID*/`lab7`) Complete the Lab 7 as instructed. In particular, you will need to properly:

a) (10 marks) Submit the file '`tweets.csv`' as instructed in Step 6.

b) (10 marks) Submit the file '`find_hashtags.py`' as instructed in Step 7.

4) (25 marks, answer to be submitted via SVN in the file *CSID*/`a3/a3q4.txt` or *CSID*/`a3/a3q4.pdf`) Consider the phrase 'mississippi shipping'. We will consider an alphabet of 26 lowercase English letters, and the space character used to separate the words, which makes the total vocabulary of 27 characters. Our main task it to create a uni-gram character model; i.e., a Markov Chain model, of this sentence, or in other words, to calculate probability estimates for each character.

a) (7 marks) What are probabilities of the 27 characters based on the given phrase if we do not use smoothing?

b) (8 marks) What are the probabilities if we use Laplace add-one smoothing method?

c) (10 marks) What are the probabilities if we use Witten-Bell smoothing method?

5) (35 marks, answer to be submitted via SVN in the file *CSID*/`a3/a3q5.txt` or *CSID*/`a3/a3q5.pdf`) Let us assume that you work on a problem of sentiment classification of Twitter data on certain topic. After analyzing a set of comments, you found two words, let us call them A and B, that seem to co-occur particularly often with positive tweets, and one word, let us call it C, that appears more often with negative tweets. You decided to work on creating a small Naïve Bayes classifier to classify tweets into positive and negative class. To test the method, you decided to use the following features in classification:

- The feature $A \in \{t, f\}$, which is set to 't' (true) if the A word appears in a tweet, and otherwise it is set to 'f' (false).
- The feature $B \in \{t, f\}$, which is set to 't' (true) if the B word appears in a tweet, and otherwise it is set to 'f' (false).
- The feature $C \in \{t, f\}$, which is set to 't' (true) if the C word appears in a comment, and otherwise it is set to 'f' (false).

The class itself is modeled using the variable $P \in \{t, f\}$, where t stands for a positive tweet, and f stands for a negative (or non-positive) tweet.

The training data is presented in the following table:

tweets	A	B	C	P
10	f	f	f	f
18	f	f	f	t
40	f	f	t	f
69	f	f	t	t
1	f	t	f	f
2	f	t	f	t
2	f	t	t	f
5	f	t	t	t
5	t	f	f	f
6	t	f	f	t
18	t	f	t	f
19	t	f	t	t
2	t	t	t	f
3	t	t	t	t

a) (15 marks) Calculate the conditional probability tables (CPTs) for the Naïve Bayes model.

b) (5 marks) Calculate P($P = t \,|\, A = f, B = t, C = f$) using the Naïve Bayes model and briefly describe what this conditional probability represents.

c) (5 marks) What is the most likely value of the class variable P for the partial configuration ($A = f, B = t, C = f$) according to the Naïve Bayes model discussed in a) and b)?

d) (5 marks) What is P($P = t \,|\, A = f, B = t, C = f$) if we use the Joint Distribution Model?

e) (5 marks) What is P($P = t \,|\, A = f, B = t, C = f$) if we use the Fully Independent Model?

Note: In assignments, always include intermediate results and sufficient details about the way the results are obtained.

6) (25 marks, answer to be submitted via SVN in the file *CSID*/`a3/a3q6.pl` or with another file extension) Write and submit a program written in Perl, Python, C, C++, or Java, named `a3q6.pl`, `a3q6.py`, `a3q6.c`, `a3q6.cc`, or `a3q6.java`, which calculates the CNG distance between two files.

The program is executed using one of the following commands:

```
perl a3q6.pl      n L file1 file2
python a3q6.py    n L file1 file2
./a.out           n L file1 file2
java a3q6         n L file1 file2
```

where n is a positive integer denoting n-gram size, L is a positive integer denoting profile length, and *file1* and *file2* are the names of two files.

The program must print a single output line to the standard output in the following format:

```
CNG for n L file1 file2: dist
```

where *dist* is a floating-point number denoting the distance between the files.

The detailed specifications of the program algorithm are as follows:

- The program will read the contents of the first file as one string, and prepend and append one space character at the beginning and at the end of file.
- All sequences of whitespace characters are replaced with a single underscore (_) character in the string.

- All character n-grams of size *n* are collected from the string, and n-gram frequencies are normalized by dividing them with the total sum of all frequencies to obtain numbers between 0 and 1.
- The n-grams are sorted by frequency starting from the most frequent ones, and the first *L* n-grams are kept as the profile with their frequencies. If several n-grams have the same frequency, they are additionally sorted lexicographically.
- The profile of *file1* is saved into a file named *file1*`.ngrams` in which every line will contain an n-ngram followed by its frequency. The n-grams are printed without spaces between characters, and there is one space betwen n-gram and the number. The lines must be sorted in the same way as in the profile.
- The above procedure of creating and saving the profile is repeated with the file *file2*, from the start.
- The CNG distance is calculated as discussed in class between the profiles, and printed according to the above specification.

Use a header comment in your solution as in the previous programming assignment.