DALHOUSIE UNIVERSITY
Inspiring Minds

Winter 2020 (Jan6-Apr6)
Faculty of Computer Science

CSCI 4152/6509 — Natural Language Processing

Assignment 2

Due: *Wednesday, Feb 26, 2020 by midnight*
Worth: 150 marks ($= 22 + 26 + 12 + 25 + 35 + 30$)
Instructor: Vlado Keselj, CS bldg 432, 902.494.2893, vlado@dnlp.ca

---

Assignment Instructions:

The submission process for Assignment 2 is mostly based on the `submit-nlp` command on `bluenose` as discussed in the lab, or in the equvalent way by using the course web site, where you need to follow 'Login' and then the 'File Submission' menu option. Some questions may have some specific different submission instructions, in which case you should follow those.

Important: You must make sure that your course files on bluenose are not readable by other users. For example, if you keep your files in the directory `csci6509` or `csci4152` you can check its permission using the command:

```
ls -ld csci6509
```
or
```
ls -ld csci4152
```
and the output must start with `drwx------`. If it does not, for example if it starts with `drwxr-xr-x` or similar, then the permissions should fixed using the command:

```
chmod 700 csci6509
```
or
```
chmod 700 csci4152
```

1) (22 marks) Complete Lab 3 as instructed. In particular, you will need to properly:

a) (3 marks) Submit the example file '`array-examples.pl`' as instructed.

b) (3 marks) Submit the example file '`test-hash.pl`' as instructed.

c) (5 marks) Submit the program file '`letter_counter_blanks.pl`' and the output file '`out_letters.txt`' (or `out_letters.txt.gz`) as instructed.

d) (6 marks) Submit the program '`word_counter.pl`' and the output file '`out_word_counter.txt`' as instructed.

e) (5 marks) Submit the program '`word_counter2.pl`' as instructed (Step 6).

Note that any program that you submit needs to compile. Even if a complete source code is given in the lab, you need to type it instead of using cut-and-paste, and you need to make

sure not to introduce any errors into the program. This follows from the lab instructions that programs must be tested before submitting.

2) (26 marks) Complete Lab 4 (GitLab and Git) as instructed. In particular, you will need to properly:

a) (5 marks) Prepare and submit via Git the file `README.md` as instructed.

b) (5 marks) Prepare and submit via Git the directory `lab5g` and your public key `id_rsa.pub` as instructed.

c) (6 marks) Prepare and submit via Git the files `explore.pl` (commits for version 1.0 and 1.1) and the Hamlet file as instructed.

d) (5 marks) Create the branch `ada-main-program` with required commits and merged later into `master` as instructed.

e) (5 marks) Create the branch `bob-function-explore` with required commits and merged later into `master` as instructed.

3) (12 marks) Complete the Lab 5 (Python NLTK 1) as instructed, and submit files using the command `submit-nlp` as instructed. In particular, you will need to properly:

a) (3 marks) Submit the file '`list_merge.py`' as instructed.

b) (3 marks) Submit the file '`stop_word_removal.py`' as instructed.

c) (3 marks) Submit the file '`explore_corpus.py`' as instructed.

d) (3 marks) Submit the file '`movie_rev_classifier.py`' as instructed.

4) (25 marks) Your solution should be in a file named `a2q4.pdf`, `a2q4.jpg`, or `a2q4.txt` and it should be submitted the `submit-nlp` command.

Your solution can be submitted as a plain-text, PDF, or image JPG file. The parts b) and c) require drawing of the curves, which you can create in any software that you like, or even draw it by hand using a ruler and take picture of it. It should be clear to read and verify that it is correct.

Suppose that a search engine returned 16 ranked results to our query, and when we checked them, the following are our judgements on their relevance:

1. relevant
2. relevant
3. relevant
4. not relevant
5. relevant
6. relevant

7. not relevant
8. not relevant
9. relevant
10. relevant
11. not relevant
12. not relevant
13. relevant
14. not relevant
15. not relevant
16. not relevant

Assuming that the total number of relevant documents in the collection is 25, do the following tasks:

a) (5 marks) Calculate precision, recall, and F-measure for the returned results.

b) (10 marks) Draw the Precision-Recall curve for these results. Show how the appropriate coordinates were calculated.

c) (10 marks) Draw the interpolated precision curve. Show how the appropriate coordinates were calculated.

5) (35 marks) Submit your solution as a file named `a2q5.txt` or `a2q5.pdf` using the command `submit-nlp`. Let us assume that you work on a problem of detecting positive microblog messages about financial markets. After analyzing a set of messages you found that three features F_1, F_2, and F_3 of messages are particularly useful in recognizing whether a message is positive or negative. You decided to work on creating a small Naïve Bayes classifier to classify messages into positive and not-positive classes. To summarize, your model uses the following features:

- The feature $F_1 \in \{t, f\}$, which is set to 't' (true) if feature F_1 is present in a message, and otherwise it is set to 'f' (false).
- The feature $F_2 \in \{t, f\}$, which is set to 't' (true) if feature F_2 is present in a message, and otherwise it is set to 'f' (false).
- The feature $F_3 \in \{t, f\}$, which is set to 't' (true) if feature F_3 is present in a message, and otherwise it is set to 'f' (false).

The class itself is represented using the class variable $C \in \{p, n\}$, where p stands for a positive message, and n stands for a non-positive message.

The training data is presented in the following table:

messages	F_1	F_2	F_3	C
8	t	t	t	p
2	t	t	t	n
42	t	t	f	p
1	t	t	f	n
4	t	f	t	p
31	t	f	t	n
22	t	f	f	p
6	t	f	f	n
1	f	t	t	p
11	f	t	t	n
3	f	t	f	p
3	f	t	f	n
1	f	f	t	p
90	f	f	t	n
4	f	f	f	p
21	f	f	f	n

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

b) (5 marks) Calculate $\mathrm{P}(C = p \,|\, F_1 = f, F_2 = t, F_3 = f)$ 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 C for the partial configuration $(F_1 = f, F_2 = t, F_3 = f)$ according to the Naïve Bayes model discussed in a) and b)?

d) (5 marks) What is $\mathrm{P}(C = p \,|\, F_1 = f, F_2 = t, F_3 = f)$ if we use the Joint Distribution Model?

e) (5 marks) What is $\mathrm{P}(C = p \,|\, F_1 = f, F_2 = t, F_3 = f)$ 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) (30 marks) You need to write and submit a program in one of the languages: Perl, Python, C, C++, or Java, according to the specifications given below. Depending on the language that you use, the program must be named either `a2q6.pl` (for Perl), `a2q6.py` (for Python), `a2q6.c` (for C), `a2q6.cc` (for C++), or `a2q6.java` (for Java). Python 2 and Python 3 versions of the program should be distinguished by the first line, which should be either '`#!/local/bin/python2`' for Python 2 or '`#!/local/bin/python3`' for Python 3 (according to the bluenose environment). The program must be submitted using the command `submit-nlp` (or via the equivalent form on the web site). The program must read the standard input, write to the standard output, and not open any files or use any other kind of interaction with the system or network.

A frequent task that we need to do when working with HTML files from internet is removal of the HTML tags and comments. You need to write a program similar to that, but in order to test it, we do not want simply to remove the tags but to hide them in a way. We want to replace any tag, such as `<a href="abc">` with the string `<............>`. Any character between delimiters `<` and `>` should be replaced with a period (`.`), except new-line characters (`\n`). Additionally, we also want to recognize HTML comments in text, which start with `<!--` and end with `-->`, and similarly replace all characters except new-line inside comments with periods. For example, `<!-- comment<> -->` should be replaced with `<!--..........-->`. In case that input contains a tag that starts with `<` but never ends, or a comment that starts with `<!--` but never properly ends, we should still replace the non-new-line characters with period from that position to the end of input.

Since tags and comments may overlap in different ways, you must follow the following principles. Always detect the leftmost start of a tag or comment: if you detect the leftmost character `<` to be followed by `!--` it must be treated as opening comment, otherwise it is an opening tag. After that, you must search for a corresponding closing string, either `-->` for a comment, or `>` for a tag, and then processing continues. It is not really important how exactly you do processing, as long as the output is exactly as specified.

Below, you can find some short examples of input and output:

```
Input:  Start <!-- in comment><!--no nesting-->123-->
Output: Start <!--.........................-->123-->
Input:  Start <tag <!--comment? --> maybe?> end
Output: Start <...................> maybe?> end
Input:  Start <tag <!--comment>? --> maybe?> end
Output: Start <..............>? --> maybe?> end
Input:  Start <tag <!--comment>? <!--> maybe?> end
Output: Start <..............>? <!--............
```

For example, if we use the following file `test1.html.txt` as input:

```
<html><body>
<h1>This is a header</h1>
Normal text <a href="some_link">link</a>.
<br>
This is a multiline <a target="_blank"
  href="this_should_all_be hidden"
 >And so on</a> link.
<br>
Start comment: <!-- >this is all<inside>
comment we can use > and < and so on
until --> :this is out of comment.
<br>
Check in browser if you want.<br>
```

we should get the following output:

```
<....><....>
<..>This is a header<...>
Normal text <.................>link<..>.
<..>
This is a multiline <..............
................................
.>And so on<..> link.
<..>
```

```
Start comment: <!--.....................
.....................................
......--> :this is out of comment.
<..>
Check in browser if you want.<..>
```

These sample input and output files (`test.html.txt` and `test1.out`) are provided in the assignment directory, and you can test your program with commands:

```
./a2q6.pl < test1.html.txt > test1.new
diff -s test1.out test1.new
```

and if there are no differences, it means that your program works correctly on this input. Additionally, if you test the number of lines and characters in both files with:

```
wc test1.html.txt test1.out
```

you will notice that both files have exactly the same lenght and the same number of lines (output):

```
 13  53 339 test1.html.txt
 13  37 339 test1.out
 26  90 678 total
```

Note: You can also change the name of the file `test1.html.txt` to `test1.html`. We added the extension `.txt` so that it does not open as an HTML file when access via a web browser.

You can find another test case `test2.html.txt` and `test2.out`, which is based on a Dalhousie course timetable page.