Winter 2017 (Jan9-Apr10)
Faculty of Computer Science

CSCI 4152/6509 — Natural Language Processing

Assignment 2

Due: *Friday, Feb 17, 2017 by midnight*
Worth: 78 marks
Instructor: Vlado Keselj, CS bldg 432, 902.494.2893, vlado@dnlp.ca
TA: Magdalena Jankowska, jankowsk@cs.dal.ca

Assignment Instructions:

The first question of Assignment 2 refers to the Lab 2, so you will need to follow the lab instructions.

For the rest of the Assignment 2 questions, similarly to Assignment 1, the solutions must be submitted using SVN. The answer files for questions not related to Lab 2 (question 2 and larger), must be located in the `a2` subdirectory of your SVN directory. This means that to solve these questions, your first commands after logging in into your bluenose account should probably be:

```
cd csci4152/CSUSERID
```

or

```
cd csci6509/CSUSERID
```

then

```
svn mkdir a2
cd a2
```

The answer files to questions 2, 3, and 4 must be in this directory, and submitted through SVN. Your answer file for question 2 should be named `a2q2.txt` and so on (each answer file name is indicated at the beginning of each question).

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

1) (18 marks, files in *csuserid*/`lab2`) Complete the Lab 2 as instructed. In particular, you will need to properly:

a) (4 marks) Submit the file 'example2.pl' as instructed.

b) (4 marks) Submit the file 'example5.pl' as instructed.

c) (5 marks) Submit the file 'task1.pl' as instructed.

d) (5 marks) Submit the file 'task2.pl' as instructed.

Notice that the examples from (a) and (b) need to compile; if a syntax error got introduced to an example program by your typing mistake or by introducing incorrect characters through copying and pasting from a pdf file, so that the example program does not compile, it will not be accepted. The lab instructions state that the programs should be tested before being submitted.

2) (10 marks, file `a2q2.txt`) List the morphological processes mentioned in the class, and briefly describe one of them with some examples.

3) (30 marks, `a2q3.txt` or `a2q3.pdf`) In this question, you are asked to do some calculations and include them either in a textual file called `a2q3.txt` or a PDF file called `a2q3.pdf`.

Let us assume that five very simple documents are given as follows:

```
d1: the dog eat homework
d2: the cat eat homework
d3: the dog and the cat eat the hot dog
d4: the dog play with the cat
d5: the cat play with the cat
```

The names `d1`, `d2`, etc. are names of the documents, and each document is simply the line of text following the name and the colon. So, the first document `d1` contains just the text:
`the dog eat homework`
Note that the terms (words) in the documents are stemmed, so they are not always grammatically correct (for example, 'eat' appears instead of grammatically correct 'eats' in `d1`).

a) (20 marks) Your first task is to calculate the document vectors using the *tfidf* weights, according to the vector space model discussed in the class. When presenting vectors, you should use all terms from the documents (9 of them) sorted in the lexicographic order.

You need to show the intermediate calculation results, in particular, the *df* and *idf* for all terms, and also *tf* and *idf* values for all terms over all documents. At the end, show clearly what are the document vectors for all five documents. For simplicity reasons, you can show all calculations with precision of only two decimal places. For the calculation of log (logarithm) function, you should assume the natural logaritm (base e).

b) (10 marks) Calculate cosine similarity between vectors for documents `d1` and `d2`, and then also between vectors for documents `d1` and `d4`. Is `d1` more similar to `d2` or `d4` according to the cosine similarity measure?

Show the intermediate calculation. You can use precision of just two decimal places, or full precision since there are not that many numbers.

4) (20 marks, `a2q4.pl`) Write and submit a Perl program named `a2q4.pl` that measures CNG distance between lines of text. The program should read the standard input using the Perl operator `<>`. The input will be provided in the following form:

```
d1: the dog eat homework
d2: the cat eat homework
d3: the dog and the cat eat the hot dog
d4: the dog play with the cat
d5: the cat play with the cat
```

We will call each line of text a 'document', since this is a toy example of larger document processing. As we can see from the example above, each line will start with document identifier, such as `d1`, followed by a colon, some arbitrary spaces and the document content, which is the rest of the line.

The program should compare each document with each other document that follows it in the order given in the input and print their CNG distance (as discussed in class). You should use character trigrams ($n = 3$) and to ensure exactly the same interpretation of the n-grams, you should use the Ngrams module over each document content.

The output format should be obvious from the following sample output (obtained from the give sample input):

```
d1 d2: 36.4444444444444
d1 d3: 94.2188998879476
d1 d4: 108.626347738663
d1 d5: 120.447882539377
d2 d3: 91.2342778616658
d2 d4: 112.566859338901
d2 d5: 93.7366103490376
d3 d4: 104.966653644337
d3 d5: 119.021744036803
d4 d5: 29.3333333333333
```

As can be seen from the sample output, you should iterate through all document pairs in the order that they appear in input, and print their CNG distances in the above format. The documents are not compared to themselves, so we do not have pairs such as `d1 d1`, and since the distance is symmetric, there is no need to print both symmetric pairs, such as `d1 d2` and `d2 d1`, but only the first one according to the input order..

Implementation guidelines:

• In order to ensure that the n-grams are interpreted in the intended way, you should use the Ngrams module, which is available in the `~prof6509/public` directory. You can use this module by including the following two lines at the beginning of your program:

```
use lib ‘echo -n ~prof6509‘.’/public’;
use Text::Ngrams;
```

The ‘use lib’ directive tells Perl to include directory `‘echo -n ~prof6509‘.’/public’` when searching for modules and libraries. This directory is evaluated to `users/marker/prof6509/public` and this will work on the `bluenose` server. If you are developing the program on your own computer, you will need to install or copy the Ngrams module there. Do not submit the Ngrams module with the assignment files to SVN.

• When reading lines of input, you can assume that document identifiers consist of non-space characters, so the regular expression `/^(\S+?):\s*(.*)$/` should match each line of input, where parentheses are used to capture the document identifier (docid) and the document content.

• The following code snippet shows how to get character trigrams from a piece of text and have them ready in an associative array:

```
$n = 3;
my $ng = Text::Ngrams->new( windowsize => $n);
$ng->process_text($text);
my %a = $ng->get_ngrams( n=>$n, normalize => 1 );
# The associative array %a contains ngrams and their frequencies.
```

• The sample input and output files are included in the assignment directory.