CSCI 3907/6907
Introduction to Statistical Natural Language Processing
Homework #2 Due October 10th, 2017 11:59pm EST
Instructor: Yassine Benajiba, yassine@gwu.edu
TA: Ali Seyfi, ali@gwmail.gwu.edu

Instructions

1. *Assignment questions 1, 2, 3, 4, 6 should be answered by all students (Undergraduates and Graduates). Question 5 should be implemented by all Graduate students, and will be considered extra credit for Undergraduates.*
2. Please submit the assignment through Blackboard
3. If you have questions about the assignment, please post them to Piazza or Blackboard.
4. Please submit questions way in advance of due date.
5. You may request a grace period of up to 7 days throughout the semester, use it wisely
6. Please include a README file that has the following aspects:
    a. Your name and email address
    b. Homework number
    c. A description of every file in your solution, the programming language used, supporting files, any additional resources used, etc.
    d. How your system operates, in detail.
    e. A description of special features (or limitations) of your system
    f. How to run your system, with a sample command line.

7. Within Code Documentation:

    a. All environment variables should be set appropriately within the program.
    b. Methods/functions/procedures should be documented in a meaningful way. This includes informative method/procedure/function/variable names as well as explicit documentation.
    c. Efficient implementation
        i. Don't hardcode things that should be variables, etc.
        ii. Choose meaningful names for your variables

8. Programming considerations

    a. The code should be written in Perl, Python, C++, or JAVA
    b. The code should compile on the SEAS machines
    c. We will not debug the code
    d. You must write the code yourself. Do not use publicly available code or copy code from any other source and do not let anyone

else look at or use your code (please refer to the Academic Integrity policy below and on the course syllabus or ask the instructor or TA if you have questions in this regard). Please check this explicitly before submitting your assignment.

e. If you wish to use any additional tools please check with the TA first by posting a question to Blackboard or Piazza.

Academic Integrity

Copying or paraphrasing someone's work (code included), or permitting your own work to be copied or paraphrased, even if only in part, is not allowed, and will result in an automatic grade of 0 for the entire assignment or exam in which the copying or paraphrasing was done. Your grade should reflect your own work. If you believe you are going to have trouble completing an assignment, please talk to the instructor or TA in advance of the due date.

==========

Assignment 2

Total points: 180 (graduate), 150 (undergrad)

Q1. Finite State Transducer [30 points]

The Soundex algorithm (Odell and Russell, 1922; Knuth, 1973) is a method commonly used in libraries and older Census records for representing people's names. It has the advantage that versions of the names that are slightly misspelled or otherwise modified (common, for example, in hand-written census records) will still have the same representation as correctly-spelled names. (e.g., Jurafsky, Jarofsky, Jarovsky, and Jarovski all map to J612). The rules are as follows:

- Keep the first letter of the name, and drop all occurrences of non-initial a, e, h, i, o, u, w, y
- Replace the remaining letters with the following numbers: b, f, p, v → 1 c, g, j, k, q, s, x, z → 2 d, t → 3 l→4 m, n → 5 r→6
- Replace any sequences of identical numbers, only if they derive from two or more letters that were *adjacent* in the original name, with a single number (i.e., 666 → 6).
- Convert to the form Letter Digit Digit Digit by dropping digits past the third (if necessary) or padding with trailing zeros (if necessary).

Write a FST to implement the Soundex algorithm.

Q2-6 Language Modeling

Language identification is the problem of taking a text in an unknown language and determining what language it is written in. N-gram models provide a very effective solution for this problem.

For training, use the English (EN), French (FR), and German (GR) texts made available from the class webpage. For test, use the file LangId.test.txt provided on the class webpage. For each of the following problems, the output of your program has to contain a list of [line_id] [language] pairs:

ID	LANG
1	EN
2	FR
3	GR

Q2. [30 points]. Bigram Letter models

- Implement a letter bigram model, which learns letter bigram probabilities from the training data. A separate bigram model has to be learned for each language (from each of the training files provided). Apply the models to determine the most likely language for each sentence in the test file (that is, determine the probability associated with each sentence in the test file, using each of the four language models).
- *Save your program as letterLangId.* (depending on the programming language used, you can use perl, java, c++, please specify the compiler version we can use to run your program)*
- *Save the output as BigramLetterLangId.out*

Q3. [30 points]. Bigram Word models – Add-One smoothing

- Implement a word bigram model, which learns word bigram probabilities from the training data. Again, a separate model will be learned for each language. Use Add-One smoothing to avoid zero-counts in the data. Apply the models to determine the language ID for each sentence in the test file.
- *Save the program as BigramWordLangId-AO.**
- *Save the output as BigramWordLangId-AO.out*

Q4. [30 points]. Bigram Word models – Good-Turing smoothing

Same as 2, but replace the add-one smoothing with Good-Turing smoothing.

- Implement a word bigram model, which learns word bigram probabilities from the training data. Again, a separate model will be learned for each language. Use Good Turing smoothing to avoid zero-counts in the data. Apply the models to determine the language ID for each sentence in the test file.
- *Save the program as BigramWordLangId-GT.**
- *Save the output as BigramWordLangId-GT.out*

Q5. [30 points]. Trigram Word models – Katz Back-off

- Implement a word Trigram model, which learns word trigram probabilities from the training data. Again, a separate model will be learned for each language. Use Katz Back-off modeling as described in chapter 4 of Jurafsky and Martin Book to avoid zero-counts in the data. Apply the models to determine the language ID for each sentence in the test file.
- *Save the program as TrigramWordLangId-KBO.**
- *Save the output as TrigramWordLangId-KBO.out*

Q6. [30 points]. Quantitative Error Analysis – A pdf report

- Accuracy: Create a table to report the accuracy of your systems for each of the homework questions comparing with the provided gold file LangId.gold.txt. The overall accuracy of your system is the percentage of correctly identified sentences/total number of sentences. You are comparing each of your *.out files with the gold file. The table has the following format.

Experimental Condition	**Overall Accuracy %**
BigramLetterLangId	
BigramWordLangId-AO	
BigramWordLangId-GT	
TrigramWordLangId-KBO	

- Confusion Matrix: For each of the systems, create a confusion matrix showing the confusability between languages. Per system, we would like to know what languages were more confusable. Therefore, for those sentences that were ***incorrectly*** identified, report the % of sentences confused with each of the languages. For example if your system BigramLetterLangId got 60% of the sentences correctly, then 40% of the sentences were incorrectly classified. Of those 40%, what percentage confused language 1 with language 2, and language 3. Accordingly, per system (experimental condition name, e.g. BigramLetterLangId), create a table with the following format, where the cell values are the % of sentences mis-classified. Note that every row in the table should add up to 100%.

Per Experimental Condition	EN	FR	GR
EN	Nothing here	% Gold French sentences identified as English sentences	
FR		Nothing here	
GR			Nothing here

- Perplexity of Test Set: For each of the 3 (4) experimental conditions in questions 1-3 (4 for graduate students), calculate the perplexity of the Test Set (LangId.test.txt) against the language model you created.
    iv. The program should take in two parameters, the test data and a LM
    v. You should produce a report with a table of 9 (12) for graduate students) scores formatted as follows

Experimental Condition	EN	FR	GR
BigramLetterLangId	Perplexity of test data against bigram letter model		
BigramWordLangId-AO			
BigramWordLangId-GT			
TrigramWordLangId-KBO			