University of Alberta
CMPUT 497/501 Fall 2019
Assignment 2

In this assignment, you will explore an interesting application of language models: automatically identifying the language of a text.

The frequency distribution of character n-grams varies between languages. Therefore, a character language model will typically assign higher probabilities (and so, lower perplexities) to text in the same language as the text it was trained on. For example, a character language model trained on English will, on average, assign a lower perplexity to an English sentence than a German sentence. In this assignment, you will use this property of language models for language identification. The algorithm is as follows: First, train a variety of language models, each using text from a different language as training data. Then, given a text, compute the perplexity of that text under each of the models. The language of the model that assigns the lowest perplexity to the text is a reasonable guess for the language of the text.

Included with this assignment is a training set of 55 files with names of the form 'udhr-.*.txt.tra'. Each such file contains part of the Universal Declaration of Human Rights (UDHR) in a variety of languages. You will use these texts to induce character n-gram language models for each of the represented languages.

Also included is a development set of files with names of the form 'udhr-.*.txt.dev'. These files each contain another, smaller, part of the UDHR from the same set of languages covered by the first set of files. Use these files to test your program during development. A final test set will be released later.

Your task is to write a Python program which, given samples of text in a variety of languages (such as the training files), identifies the languages of each of a given set of documents (such as the development or test files). Your program should use the method described above: train a language model for each language from its sample, and for each document to be classified, determine its perplexity using each language model.

Your program should be able to train three different types of language model: unsmoothed, smoothed with add-one (Laplace) smoothing, or smoothed with linear interpolation smoothing. The user of the program will select the type of model trained at run time using a single command line argument: "--unsmoothed", "--laplace", or "--interpolation". For interpolation, the lambda coefficients should be set using the deleted interpolation algorithm, as described in the J&M textbook (chapter 8, pages 15-16, 3rd edition) applied to the training files.

For each type of model, you must tune the parameter *n*, which controls the size of the context used by the model. The value of *n* may be different for each type of model, but, for each type of

model, the value of *n* should be the same across all languages. Thus, you have exactly three values of *n* to tune.

The output of your program should be in the following tab-separated format:

- Each line should correspond to one of the files to be classified;
- these files should appear in alphabetical order.
- The name of the test files should be followed by the name of the training file which corresponds to the best guess (i.e., which file induced the model that had the lowest perplexity for that test file), followed by the perplexity, followed by the value of the *n* parameter, with the 4 fields separated by tabs. For example, one line might read as follows:

  udhr-deu_1996.txt.dev udhr-eng.txt.tra 8.44 3

Once the test set is released, you should produce 3 files containing the language classification output of the test files corresponding to the following smoothing methods: (a) no smoothing, (b) add-one (Laplace) smoothing, and (c) linear interpolation smoothing. The names of the files should be 'results_test_unsmoothed.txt', 'results_test_add-one.txt', and 'results_test_interpolation.txt'. Include also the corresponding development set outputs, with the names 'results_dev_unsmoothed.txt', 'results_dev_add-one.txt', and 'results_dev_interpolation.txt'.

You must submit a report in the PDF format. The max. length of the report is 2 two-column pages. Your report should include the following:

- description of the method, and how you implemented it, and any issues that you encountered
- how you tuned your parameters (and what values you decided to use),
- discussion of the relative performance of the smoothing variants and n-gram settings
- error analysis
- citations all external sources
- for team submissions, explanation of who performed which tasks

Your submission should consist of a single zip file, containing the following files:

- the 6 required output files
- your report, in PDF format
- all the code that you wrote for the assignment
- a README file in plain text format describing how to run your code (e.g. how to choose the type of model), as well an example of how to run your program.

You will be evaluated on the basis of the correctness, efficiency, and documentation of your code, its performance relative to other submissions, the quality of your report and README, and the inclusion of all required files.

The following sequence of steps is the suggested procedure for completing this assignment:

1. Write code which, given a value for n, generates an unsmoothed n-gram language model over characters. You have already seen in the NLTK tutorial how to generate an unsmoothed bigram language model over words -- this may be helpful.
2. The second step will build upon this code with smoothing. Modify the code you wrote in the first step to incorporate the smoothing techniques (and apply them depending on the command line parameter), in order to prevent your models from assigning zero probabilities to unseen n-grams. Remember to provide a command line parameter to allow the user to choose the smoothing method.
3. Write your program, named 'langid.py', that loops over the files listed in the input file, assigns each a perplexity with each language model, and produces output as described above.
4. Produce output files for the dev and test files.
5. Write your README and report.
6. Gather all the required files, compress them into a zip file, and submit it.

Assignment submission guidelines:

- You are allowed to work with a partner. Make only one submission per team. Provide the names of the team member at the beginning of the README file.
    - If you work with a partner, provide a submission comment on eClass along with your submission giving their name and CCID.
- Name your zip file using your CCID, or both CCIDs for group submissions. Upload your zip file using the button on the bottom of the assignment page.