ANLP Assignment 1 2015

due: Monday 19 October, 3pm, electronic submission (see end of instructions)

1 Overview

In this assignment, you will use language modeling to detect what language a document is written in. Specifically, you will write a Python program that does the following:

- builds a trigram language model over characters (not words!): reads in a text file, collects counts for all character 3-grams, estimates probabilities, and writes out the model into a file.
- generates random output, according to its probabilistic model.
- reads in a test document, applies the language model to it, and outputs its perplexity.

and you will need to write a report that explains some of what you did and answers some questions.

The goals of this assignment are to help you better understand some of the issues involved in language modeling, and to give you practice with (a) programming, (b) working with real language data, and (c) being able to clearly explain/justify decisions you have made and describe results you have obtained and the conclusions that can be drawn from them. We don't just care that you got the right answer, we also care that you are able to explain how and provide evidence justifying your answer.

2 Working with others (and not)

The assignments for this class are intended to be done in *pairs*: by working with another student, you can discuss ideas and work things out together. Ideally, try to find a partner with a different skill set to your own, although this may not be possible in all cases. You are free to choose your own partner, but we ask you to work with different partners for each assignment.

You are free to discuss any aspects of the assignment with your partner and divide up the tasks however you wish; however we encourage you to collaborate on each part rather than doing a strict division of tasks, as this will enable better learning for both of you.

You are also free to discuss high-level concepts and general programming questions with others in the class; however you may NOT share code or answers directly with other groups. For example, it is ok to suggest using a particular library function to help solve a task; it is not ok to send (or describe) a code snippet showing exactly how you used that function on what arguments. Your code and report must be your own group's work. Any key ideas obtained from outside sources should be appropriately cited, and direct quotes must also be included in quotation marks. If you re-use code snippets from outside sources, these should also be clearly marked and the source cited.

Please see the School's guidance on plagiarism and academic misconduct for further information:

`http://web.inf.ed.ac.uk/infweb/admin/policies/guidelines-plagiarism`

3 Further specifiation

3.1 Data

The data we use for this project is part of the Europarl corpus, which requires a license to use. The University has a license, so to download the data you must be inside the University intranet. You may use the data for your University work but may not distribute it to others.

From a University computer, the data is available at:
`http://www.inf.ed.ac.uk/teaching/courses/anlp/data/assignment1-data.tgz`

On a DICE machine you can download and unzip it with Archive Manager. You should find that the data consists of:

- `training.en` - English training data
- `training.es` - Spanish training data
- `training.de` - German training data
- `test` - test document

3.2 Code

To help students who are new to programming, we've provided a very small amount of code that may help you get started with the tasks specified below. The code is available here:
`http://www.inf.ed.ac.uk/teaching/courses/anlp/hw/code/asgn1-helper.py`

You are not required to use the provided code, and if you are an experienced programmer you may prefer not to.

3.3 Tasks

1. (10 marks) Before starting to code, you should always have an idea of what your program's expected output should be, so you can check that your implementation is correct. In the last part of this assignment, you'll be asked to compute the perplexity of some real language data under a language model that's also computed from real data. But to test your code, you'll want a simple test case that you've done by hand.

   Consider the following trigram character language model, where `[` and `]` are the start- and end-of-sequence symbols, and probabilities are written as $P(c_i \,|\, c_{i-2}, c_{i-1})$. So, $P(\mathtt{a} \,|\, \mathtt{[b})$ means the probability of the character `a` given that the immediately preceding character is `b` and the one before that is `[`.

$$\begin{array}{lll}
P(\mathtt{a} \,|\, \mathtt{[[}) = 0.2 & P(\mathtt{b} \,|\, \mathtt{[[}) = 0.8 & P(\mathtt{]} \,|\, \mathtt{[[}) = 0.0 \\
P(\mathtt{a} \,|\, \mathtt{[a}) = 0.2 & P(\mathtt{b} \,|\, \mathtt{[a}) = 0.7 & P(\mathtt{]} \,|\, \mathtt{[a}) = 0.1 \\
P(\mathtt{a} \,|\, \mathtt{[b}) = 0.15 & P(\mathtt{b} \,|\, \mathtt{[b}) = 0.75 & P(\mathtt{]} \,|\, \mathtt{[b}) = 0.1 \\
P(\mathtt{a} \,|\, \mathtt{aa}) = 0.4 & P(\mathtt{b} \,|\, \mathtt{aa}) = 0.5 & P(\mathtt{]} \,|\, \mathtt{aa}) = 0.1 \\
P(\mathtt{a} \,|\, \mathtt{ab}) = 0.6 & P(\mathtt{b} \,|\, \mathtt{ab}) = 0.3 & P(\mathtt{]} \,|\, \mathtt{ab}) = 0.1 \\
P(\mathtt{a} \,|\, \mathtt{ba}) = 0.25 & P(\mathtt{b} \,|\, \mathtt{ba}) = 0.65 & P(\mathtt{]} \,|\, \mathtt{ba}) = 0.1 \\
P(\mathtt{a} \,|\, \mathtt{bb}) = 0.5 & P(\mathtt{b} \,|\, \mathtt{bb}) = 0.4 & P(\mathtt{]} \,|\, \mathtt{bb}) = 0.1
\end{array}$$

What is the perplexity of the sequence `[[abaab]` under this model? Show your work.

2. (10 marks) For the three languages we gave you, the language identification task is too easy if we include the entire character set, because there are some characters that only occur in one of the languages. So, we will ask you to preprocess the data to make things a bit more interesting.

   Write a function called `preprocess_line` that takes a line of text (string) as an argument. It should return a new string which removes all characters from the line that are not in the following set: characters in the English alphabet, space, comma, or period. (That is, remove characters with accents and umlauts and the other punctuation marks). Your function should also lowercase all remaining characters and convert all numerals to '0'.

   Depending on how you write the rest of your program, you may do slightly more in this function than just what we have specified. If so, add a comment to your code explaining what else you did and why.

   You should use this function in building your language model and scoring the test document; i.e., your language model should be over n-grams of lowercase letters plus space, comma, and period.

   To get credit for this task, include the code for this function in your report.

3. (35 marks) Now write code to build the trigram character language model. You will need to read in a training file, collect counts, estimate probabilities, and write the model probabilities into a file.

   In your report, describe the method you used to estimate the probabilities, and give a brief justification for why you used that method. Your description should include, but not be limited to, an equation. You should also explain any simplifying assumptions you have made.

   Also include in your report an excerpt of the language model for English, displaying all n-grams and their probability with the two-character history $t\ h$. Explain what you would expect to find and why. Do the results match your expectations? Pick one other part of the language model (for English or another language) that you can make a prediction about, and discuss whether your prediction is borne out.

   Hint 1: In deciding on an estimation method, consider what you will need to do in the rest of the assignment, and make sure your method will work for what you need to do. You can do well on this question by implementing a simple method, as long as it works for the other tasks you will need to do (see also Section 4).

   Hint 2: Consider what kinds of data structures will be useful for storing your counts and probabilities. For example, you might want to use a dictionary with an entry for each possible history, where the value stored is the distribution over next characters given that history. (You'll then need to decide what kind of structure that distribution is!)

4. (10 marks) Extend your program so that you can generate random output from the estimated language models.

In your report, include 300 characters of random output for each of the three training languages.

5. (20 marks) Extend your program to read in a test document and compute (and output) its perplexity under the estimated language models. To test your code, you should also write a function that reads in a language model (in the same format you used to print out the LM you computed for Question 3). Then you can make a test file containing the LM from Question 1, read it in, and use it to compute the perplexity of the data from Question 1. Make sure your program gets the same result you computed by hand!

   What is the perplexity of the test document we provided under each of the three language models? How can you use these numbers to determine the language of the test document without looking at it? Would a unigram or bigram language model work (as well/at all) for the language ID task? Why or why not?

   Suppose we ran your program on a new test document and told you the perplexity under your English LM. Would this be enough for you to determine if the document is written in English? Why or why not?

6. (15 marks) Extend your work in some other way. Explain briefly what additional question(s) you decided to address, and what results you got.

   As noted in the marking guidelines (see below), this part of the assignment should be considered "extra", that is, going beyond the basic requirements of the assignment by implementing more complex methods, comparing different methods or models, making a more in-depth exploration of whichever method you choose, or writing more general code to do some of these things.

   IMPORTANT: you will only get credit for this part if your basic method is correct. Do not attempt this question unless you have completed all the previous questions, and are sure that you've done a good job on those. This question is intended to challenge students who already feel solid on the basic material in the course, and want to go further. If you are not in that position, it is not worth spending time on this question; instead you should focus on revising other material in the course. Also note that the time it takes to answer this question is not necessarily proportional to how many extra marks you'll get, though it will help you learn more!

4 Marking scheme

Assignments will be marked on the usual British scale:

Numeric mark	Equivalent letter grade	Approximate meaning
< 40	F	fail
40-49	D	poor
50-59	C	acceptable
60-69	B	good
70-79	A	very good/distinction
80-100	A*	excellent/outstanding/high distinction

You can do well (70-79) by:

- Completing all tasks and providing working solutions, even if they are simple.
- Answering all questions correctly, clearly and concisely.
- Including brief but clear comments in your code indicating what each function does, including what its arguments and return values are. See the comments inside triple quotes in the lab code for examples—you do not need to comment nearly every line as we did, but a few well-placed comments explaining what each block is doing can be useful.

We will *not* be assigning any marks based on the quality of your code (provided it works), since many students are still just learning. However we do insist that you turn in your code (a) to discourage plagiarism, and (b) since it may help us provide useful feedback.

5 Submitting your assignment

Only one student in your pair needs to submit, but make sure both student's ID numbers (s...) are included in the report!

Submit your assignment using the following command on a DICE machine:
`submit anlp 1 <file1.py> <file2.pdf>`
where you replace `<file1.py>` with the name of the file (or files, you can add more filenames if needed) with your python code and `<file2.pdf>` with the name of the file with your report.

Notes:

- Your report *must* be either a .pdf document or .txt (plain text) file, please convert other files types (eg Word documents) into .pdf format before submitting.
- You can submit more than once. Identically named files will overwrite earlier submitted versions. All submissions are timestamped, so do not submit anything after the deadline—it will be considered late, and receive a mark of 0. [Even if you submitted an earlier version, we won't be able to give you credit for that one if it's overwritten!]
- Make sure you **save** your code and report **before** submitting, and that you go through the whole submission process (you may need to answer some y/n questions) and get a message in the terminal that submission succeeded.