ANLP Assignment 1 2020

due: Tuesday, 20 October, 4pm, via git submission (see section 1.4)

1 Overview

1.1 Tasks

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

- build a trigram language model over characters (not words!): read in a text file, collect counts for all character 3-grams, estimate probabilities, and write out the model to a file.
- generate random output according to a similar model provided by us (which you'll need to read in from a file).
- read in a test document and compute its perplexity according to one of these models.

You'll also need to write a short description of what you did and why, and answer some questions.

1.2 Motivation and goals

By completing this assignment, you will practice skills that are important for doing NLP, and for other research in Informatics. This assignment will allow you to demonstrate that you:

- understand some of the concepts and issues involved in language modeling and working with real language data;
- can implement a basic Python program; and
- have some of the skills needed to write a good scientific report. These include being able to clearly explain/justify decisions you have made, describe results you have obtained, and draw appropriate conclusions 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.

1.3 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 may choose your own partner, but you must work with different partners for each assignment.

You may discuss any aspects of the assignment with your partner and divide up the tasks however you wish; but 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 may also discuss high-level concepts and general programming questions with others in the class; but you may **not** share your specific solutions, answers or code directly with other groups. Your code and report must be your own group's work. Any key ideas from outside sources should be appropriately cited, and direct quotes must use quotation marks. If you re-use code snippets from outside sources, these should also be clearly marked and the source cited.

See the School's guidance on academic misconduct for more details:

`http://web.inf.ed.ac.uk/infweb/admin/policies/academic-misconduct`

You must take reasonable measures to protect your assessed work from unauthorised access. For example, you may not store or post your work anywhere in a way that could be accessed by other students (except your partner).

1.4 Submitting your assignment

1.4.1 What to submit

Both members of your pair will receive the same mark and you are both responsible for the work, so make sure you have agreed on the final submission before submitting it. Your pair's submission should include (as separate files—do **not** zip them together):

- A single report called `a1_report.pdf`, with both student's ID numbers (s...) included at the top. Include *only* your ID numbers, not your names, to help avoid unconscious bias by the marker. Convert other files types (eg Word documents) into .pdf format before submitting.
- All of your code, as .py file(s). Do not submit .ipynb files, convert if needed.

1.4.2 How to submit

As discussed in Lab2a, we recommend strongly that you use the master branch of your shared repo only for *agreed* commits, that is, commits that you and your partner are agreed on.

While you are working on the assignment, you can structure your code and directories however you like. However, when you wish to submit, you must do as follows so that we can automatically find your submitted assignment.

For submission: Push a top-level directory named `submission` to the master branch, which must contain the files described above under "What to submit."

That is, if we clone your repo and type `ls`, we should immediately see a directory called `submission` with your submitted files in it. (There might be other top-level files or directories too, we will ignore them.) We will mark the contents of the `submission` directory from the last commit before the submission deadline. (See below regarding late submissions).

1.4.3 Resubmitting and late submissions

You may resubmit as many times as you like before the deadline. However, we will download all submissions shortly after the assignment deadline to begin marking them. Therefore, following School policy, we will not accept re-submissions of any assignments already submitted before the deadline. We will only accept late submissions from pairs who have not previously submitted.

Late submissions will be penalized unless you have an approved extension. Do **not** contact course staff directly about extensions; you must follow the instructions given on this web page:

`http://web.inf.ed.ac.uk/infweb/student-services/ito/admin/coursework-projects/late-coursewo`

2 Assignment specification

2.1 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 get the data you must be inside the University intranet**: either by using Remote Desktop or a VPN. You may use the data for your University work but may not distribute it to others.

Using DICE Remote Desktop or a VPN, the data is available at:
`http://www.inf.ed.ac.uk/teaching/courses/anlp/data/assignment1-data.tgz`

Download and unzip the data (e.g. using Archive Manager on DICE). It will consist of:

- `training.en` - English training data
- `training.es` - Spanish training data
- `training.de` - German training data
- `model-br.en` - a pre-trained language model file
- `test` - test document

2.2 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.

To get the code (and eventually submit your assignment), you'll need to clone the GitHub repo that we have created for you and your partner. You should have received a email from GitHub on behalf of `@htInEdin` inviting you to a GitHub team. The team consists of you, your partner, and and `@htInEdin` (i.e., Henry S. Thompson, who is making all this happen).

Your invitation includes a *team name*, which consists of the letter 'a' followed by some digits.

To clone your repo and get the support code, replace `aNNN` with your team name in the following command:

```
$ git clone https://github.com/ANLP2020/a1_aNNN.git anlpa1
$ cd anlpa1
$ ls -R
```

You should see a `README.md` file, these instructions and a `code` directory containing a single python file called `asgn1-helper.py`.

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

For this assignment, you must implement the solutions yourself, and should not use existing language modelling toolkits. You may use other support code provided you cite the source.

2.3 Tasks

0. (0 marks; warmup exercise) This task is not worth any marks, but you should make sure you have completed it correctly before proceeding. Since it is not part of the final mark, feel free to discuss with other students.

   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 (at least!) a simple test case that you've done by hand.

   Consider the following trigram character language model, where `#` is a symbol used to mark both the start and end of a sequence, and probabilities are written as $P(c_i \mid c_{i-2}, c_{i-1})$. So, $P(\texttt{a} \mid \texttt{\#b})$ means the probability of the character `a` given that the immediately preceding character is `b` and the one before that is `#`.

$P(\texttt{a} \mid \texttt{\#\#}) = 0.2$	$P(\texttt{b} \mid \texttt{\#\#}) = 0.8$	$P(\texttt{\#} \mid \texttt{\#\#}) = 0.0$
$P(\texttt{a} \mid \texttt{\#a}) = 0.2$	$P(\texttt{b} \mid \texttt{\#a}) = 0.7$	$P(\texttt{\#} \mid \texttt{\#a}) = 0.1$
$P(\texttt{a} \mid \texttt{\#b}) = 0.15$	$P(\texttt{b} \mid \texttt{\#b}) = 0.75$	$P(\texttt{\#} \mid \texttt{\#b}) = 0.1$
$P(\texttt{a} \mid \texttt{aa}) = 0.4$	$P(\texttt{b} \mid \texttt{aa}) = 0.5$	$P(\texttt{\#} \mid \texttt{aa}) = 0.1$
$P(\texttt{a} \mid \texttt{ab}) = 0.6$	$P(\texttt{b} \mid \texttt{ab}) = 0.3$	$P(\texttt{\#} \mid \texttt{ab}) = 0.1$
$P(\texttt{a} \mid \texttt{ba}) = 0.25$	$P(\texttt{b} \mid \texttt{ba}) = 0.65$	$P(\texttt{\#} \mid \texttt{ba}) = 0.1$
$P(\texttt{a} \mid \texttt{bb}) = 0.5$	$P(\texttt{b} \mid \texttt{bb}) = 0.4$	$P(\texttt{\#} \mid \texttt{bb}) = 0.1$

What is the perplexity of the sequence `##abaab#` under this model? (You don't need to add anything to the report for this warmup exercise.)

1. (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 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, digits, or the '.' character. (That is, remove characters with accents and umlauts and the other punctuation marks). Your function should also lowercase all remaining characters and convert all digits to '0'.

   In the remainder of your program, you should treat each line as a separate sequence for modelling purposes (rather than treating the entire input file as a single sequence).

   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 do *not* need to justify the steps we explicitly asked for.

   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 a few other characters as noted.

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

2. (10 marks) We have provided a pre-trained language model in the file **`model-br.en`**. This model was trained on a corpus of English, but not necessarily the same one we gave you.

   Each line of this file lists three characters followed by a tab and a number. The number represents the model's estimated probability of the third character given the previous two. For example, one of the lines looks like this (where `<tab>` is a tab character):

   ```
   and<tab>2.507e-01
   ```

   which means that for this model, $\hat{P}(\texttt{d} \mid \texttt{a}, \texttt{n}) = 0.2507$. The file is alphabetical, so at the beginning you will see a lot of trigrams that include spaces and punctuation characters.

   By looking at the language model probabilities in this file, can you say anything about the kind of estimation method that was used? You can't be completely certain without seeing the corpus, but based on your knowledge of English and of different estimation methods, you should be able to narrow down the possibilities.

add alpha smoothing?

Write a paragraph or so explaining which method(s) might have been used, and say what evidence your guess is based on.

Note: We designed our LM file for simplicity, not space efficiency. Real LM files are far more compact but far harder to read!

3. (35 marks) Now write code to build your own 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 briefly justify 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. Please see the Marking Guide below for hints about what is considered a good explanation; most pairs should only need 1-2 paragraphs here. Your report need **not** describe how you implemented the method in your code, but please do include comments in the code itself indicating what you did.

   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 $n\ g$. Explain what you would expect to find and why. Do the results match your expectations?

   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 3).

   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. (15 marks) Write a function or method called `generate_from_LM` that uses a language model to generate random output sequences. That is, the sequences should be generated according to the probabilities in the language model.

   Using this code, generate 300 characters of random output for each of two different models: (a) the model you estimated from the English training data, and (b) the model in `model-br.en` (you will need to write code to read the model in from the file).

   In your report, include the output from each model and explain briefly what your code does to generate a sequence given an LM. Your explanation could be in English or pseudocode, but should be clear and concise. Also comment *briefly* (max 1 paragraph) on any differences you see between the two sequences you generated, and what seems likely to have caused them.

   Hint: You can do most of this question before you have implemented your own model.

5. (15 marks) Extend your program to read in a test document and compute (and output) its perplexity under the estimated language models.

What is the perplexity of the test document we provided under each of the three language models you estimated from the training documents? How can you use these numbers to guess the language of the test document without looking at it?

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?

Hint: How will you know if you've implemented perplexity correctly? First, consider what is a reasonable value to expect (look back over the Entropy slides as a starting point). Second, consider how to test your code. One way would be to create a test file with the LM from Question 2.3, read it in, and use it to compare your program's perplexity result to your hand computation from Question 2.3. However, a single test case is rarely enough. Think carefully about other tests you could use.

6. (15 marks) Devise a further question you would like to explore and extend your work in order to address that question. Explain briefly what the question was, how you addressed it, and what results you got.

   As noted in the marking guidelines (below), if you do a good job up to question 5, with only minor errors, you will still be getting an A. This question should be considered **extra**, beyond what is expected of most students. The time it takes to answer this question is not proportional to how many extra marks you'll get, though it will help you learn more, **if** you already fully understand the previous parts. Also, just implementing a more complicated method is unlikely to gain you many marks. I am really looking for evidence here that you have thought about the issues involved in language modelling and come up with an interesting question based on that.

   WARNING: you will only get credit for this part if your basic method is correct and clearly explained. This question is intended to challenge students who already feel solid on the basic material in the course, and want to go further. **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. Also, **do not** attempt this question if you have personally spent 16 or more hours on the assignment already. The maximum number of marks awarded for this question last year was 8, or less than 1% of your overall course mark. Unless you really whizzed through the earlier questions, please stop now and spend your time and energy revising other course materials or getting more sleep. Both are likely to have a much bigger impact on your final mark.

3 Marking and how to do well

3.1 The British marking scale

Assignments will be marked on the usual British scale:

Numeric mark	Equivalent letter grade	Approximate meaning
< 40	F	fail
40-49	D	sufficient for Diploma
50-59	C	good; sufficient for MSc
60-69	B	very good
70-79	A3	excellent/distinction
80-100	A1, A2	outstanding/high distinction

Please note typical specifications for marks above 70 across the University:

A1 90-100 Often faultless. The work is well beyond what is expected for the level of study.

A2 80-89 A truly professional piece of scholarship, often with an absence of errors.
As 'A3' but shows (depending upon the item of assessment): significant personal insight / creativity / originality and / or extra depth and academic maturity in the elements of assessment.

A3 70-79
Knowledge: Comprehensive range of up-to-date material handled in a professional way.
Understanding/handling of key concepts: Shows a command of the subject and current theory.
Focus on the subject: Clear and analytical; fully explores the subject.
Critical analysis and discussion: Shows evidence of serious thought in critically evaluating and integrating the evidenced and ideas. Deals confidently with the complexities and subtleties of the arguments. Shows elements of personal insight / creativity / originality.
Structure: Clear and coherent showing logical, ordered thought.
Presentation: Clear and professional with few, relatively minor flaws. Accurate referencing. Figures and tables well constructed and accurate. Good standard of spelling and grammar.

3.2 What are we looking for?

Write your answers to each question separately (with question number) in your report. Different people will mark different parts, so please try to make each answer stand alone. If need be you can refer back or forward to your answers to other questions, but try to keep answers as self-contained as possible.

To do well on this assignment (70-79), focus on doing the following:

- Complete up to task 5 and provide working solutions, even if they are simple. You can mention in 1-2 sentences if you think there is a better solution and why, but you will not get extra marks for a detailed explanation of a complex method you did not implement.

- Answer the questions correctly, clearly and concisely.
- Include 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.
- Describe the results of your work, not a history of it. So, avoid descriptions like this one:

  > "We implemented [Method 1], which is formulated as [equation], because [reason]. [Considerably more explanation of Method 1 here]. However, we then realized that [Method 1] wouldn't work [reasons here], so instead we ended up using [Method 2]. [Explanation of Method 2]."

  This is not appropriate style for a scientific report, and it is confusing. The reader initially thinks you used one method, but then later discovers you actually did something else. The reader (whether a marker reading your assignment or a scientist reading a research paper) does not want to know the history of your ideas. The reader wants to know what you actually ended up doing, and why. The above explanation should be rewritten as follows:

  > "We used [Method 2] to implement probability estimation in this assignment, because [Method 1] does not work [reasons here], and [other reasons for using Method 2]. [Explanation of Method 2]."

- Explain the terms in your equations, and say what they mean within the context of this report (if appropriate, give actual values). When you are doing an assignment, this shows the marker that you actually understand the equation yourself. When you are writing a longer paper or thesis, in many cases the reader will not actually know ahead of time what the terms in your equations mean so explaining them also helps the reader understand.
- Unless you are explicitly asked to, do not include code in your report, and avoid detailed explanations of *how* you implemented things (in terms of data structures, iteration, etc). Focus on *what* you implemented (what does your code do) and *why*. You should demonstrate that you understand the conceptual issues, have done something reasonable where there are choices involved, and can justify why your choices are reasonable. You may also have some arbitrary or trivial choices to make, and in this case it is probably not worth mentioning those in the report. Honing your judgment about what is important and what isn't is an important academic skill and this is a good chance to start working on it.

We will **not** be assigning any marks based on the quality or style 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.