USC

CSCI 544 — Applied Natural Language Processing

Coding Exercise 1

Due: Monday, September 7, at 23:59 Pacific Time (11:59 PM)

This assignment counts for 5% of the course grade.

Assignments turned in after the deadline but before Thursday, September 10 are subject to a 20% grade penalty.

Update 2020-08-25: I added notes based on the discussion in class this morning, and also replaced the development data on Blackboard. You may use either the old or the new development data, and you should get similar results. The submission script on Vocareum will use the new development data, so only the new data will give you exactly the same results as the Vocareum submission script.

Overview

Person names in the English language typically consist of one or more forenames followed by one or more surnames (optionally preceded by zero or more titles and followed by zero or more suffixes). This situation can create ambiguity, as it is often unclear whether a particular name is a forename or a surname. For example, given the sequence *Imogen and Andrew Lloyd Webber*, it is not possible to tell what the full name of Imogen is, since that would depend on whether *Lloyd* is part of Andrew's forename or surname (as it turns out, it is a surname: Imogen Lloyd Webber is the daughter of Andrew Lloyd Webber). This exercise explores ways of dealing with this kind of ambiguity.

You will write a program that takes a string representing the names of two persons (joined by *and*), and tries to predict the full name of the first person in the string. To develop your program, you will be given a set of names with correct solutions: these are not names of real people – rather, they have been constructed based on lists of common forenames and surnames. The names before the *and* are the first person's forenames, any titles they may have, and possibly surnames; the names after the *and* are the second person's full name. For each entry, your program will output its best guess as to the first person's full name. The assignment will be graded based on accuracy, that is the number of names predicted correctly on an unseen dataset constructed the same way.

Data

A set of development and test data is available as a compressed ZIP archive on Blackboard. The uncompressed archive contains the following files:

- A test file with a list of conjoined names.
- A key file with the same list of conjoined names, and the correct full name of the first person for each example.
- Three lists of name frequencies from U.S. census data (these lists are available on the web, and included in the package for your convenience).
    - Frequency of female first names from the 1990 census
    - Frequency of male first names from the 1990 census

    - Frequency of surnames from the 2010 census
- A readme and license file, which will not be used for the exercise.

The submission script will run your program on the test file and compare the output to the key file. The grading script will do the same, but on a different pair of test and key files which you have not seen before.

Program

You will write a program called `full-name-predictor.py` in **Python 3** (Python 2 has been deprecated), which will take the path to the test file as a command-line argument. Your program will be invoked in the following way:

```
> python full-name-predictor.py /path/to/test/data
```

The program will read the test data, and write its answers to a file called `full-name-output.csv`. The output file must be in the same format of the key file.

Submission

All submissions will be completed through Vocareum; please consult the instructions for how to use Vocareum.

Multiple submissions are allowed; only the final submission will be graded. Each time you submit, a submission script is invoked, which runs the program on the test data. Do not include the data in your submission: the submission script reads the data from a central directory, not from your personal directory. You should only upload your program file to Vocareum, that is `full-name-predictor.py`; if your program uses auxiliary files (for example, lists of common names), then you must also include these in your personal directory.

You are encouraged to **submit early and often** in order to iron out any problems, especially issues with the format of the final output.

The output of your program will be graded automatically; failure to format your output correctly may result in very low scores, which will not be changed.

For full credit, make sure to submit your assignment well before the deadline. The time of submission recorded by the system is the time used for determining late penalties. If your submission is received late, whatever the reason (including equipment failure and network latencies or outages), it will incur a late penalty.

Grading

After the due date, we will run your program on a unseen test data, and compare your output to the key to that test data. Your grade will be the accuracy of your output, scaled to the output of a predictor developed by the instructional staff (so if, for example, that predictor has an accuracy of 90%, then an accuracy of 90% or above will receive full credit, and an accuracy of 81% will receive 90% credit).

Notes

- Even the best models are likely to fall short of 100% accuracy; the reason is that some names are highly ambiguous between forenames and surnames. To take two examples from the development data, both *May* and *Howard* are reasonably likely as either forenames and surnames; as it turns out, *Sarah May and…* happen to be two forenames, whereas *Ryan Howard and…* happen to be a forename and a surname. It is difficult to predict this based on distributional properties alone.
- You are allowed to use the name lists included in the package. While these are also available on the web, you **cannot** use any code that fetches these lists (or anything else) from the web at runtime; rather, you should use the local copy. You may also find it useful to preprocess these lists into a form that is more useful for your program. Whatever you do, if your program requires any auxiliary files such as name lists, you must include these in your Vocareum submission.

- You may use machine learning (if you wish), but the kind of learning that can be performed at runtime is limited by the fact that the program you submit is only provided with the test data. Any learning that uses the key file must be done separately, and the outcome (for example, parameter values) needs to be stored and given to the program that is submitted.
- While there are many properties that can be learned from the key file, some are better to **not** learn from it – for example, the distribution of forenames and surnames. As explained in the overview section, names in the development data were sampled from name lists. While it is possible to learn a distribution of names from the key file, such a distribution will be less accurate than learning the distribution from the source lists provided in the readme file. Other properties (for example, the distribution of single and double names) can be estimated from the key file, with a caveat that for some names in the data, we may not be able to determine conclusively whether they are forenames or surnames.
- There is no list of titles corresponding to the lists of forenames and surnames. The best we can do with titles is to guess what the possible titles are based on the given data.

Collaboration and external resources

- This is an individual assignment. You **may** discuss ideas for the solution with other students in class or on class message boards, but you **may not** work in teams or share any code with other students. You must be the sole author of 100% of the code you turn in.
- You may not look for solutions on the web, or use code you find online or anywhere else.
- You may not download the data from any source other than the files provided on Blackboard, and you may not attempt to locate the test data on the web or anywhere else.
- You may use packages in the Python Standard Library. You may not use any other packages.
- You may use external resources to learn basic functions of Python (such as reading and writing files, handling text strings, and basic math), but the extraction and computation of models must be your own work.
- Failure to follow the above rules is considered a violation of academic integrity, and is grounds for failure of the assignment, or in serious cases failure of the course.
- We use plagiarism detection software to identify similarities between student assignments, and between student assignments and known solutions on the web. **Any attempt to fool plagiarism detection, for example the modification of code to reduce its similarity to the source, will result in an automatic failing grade for the course.**
- Please discuss any issues you have on the Piazza discussion boards. Do not ask questions about the assignment by email; if we receive questions by email where the response could be helpful for the class, we will ask you to repost the question on the discussion boards.