CSCI 544 — Applied Natural Language Processing

Coding Exercise 2

Due: Monday, September 14, 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 17 are subject to a 20% grade penalty.

Overview

In this assignment you will write a very simple lemmatizer, which learns a lemmatization function from an annotated corpus. The function is so basic I wouldn't even consider it machine learning: it's basically just a big lookup table, which maps every word form attested in the training data to the most common lemma associated with that form. At test time, the program checks if a form is in the lookup table, and if so, it gives the associated lemma; if the form is not in the lookup table, it gives the form itself as the lemma (identity mapping).

The program performs training and testing in one run: it reads the training data, learns the lookup table and keeps it in memory, then reads the test data, runs the testing, and reports the results. The program output is in a fixed format, reporting 15 counters and 5 performance measures. The assignment will be graded based on the correctness of these measures.

Data

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

- Three corpus files in universal dependencies format, with the file names indicating whether the file is for development, training, or testing (only the training and testing files will be used in the exercise).
- Additional readme and license files, which will not be used for the exercise.

The submission script will learn the model from the training data, test it on the test data, output the results, and compare them to the known solution. The grading script will do the same, but on training and test data from a different language.

Program

You will write a program called `lookup-lemmatizer.py` in **Python 3** (Python 2 has been deprecated), which will take the paths to the training and test data files as command-line arguments. Your program will be invoked in the following way:

```
> python lookup-lemmatizer.py /path/to/train/data /path/to/test/data
```

The program will read the training data, learn a lookup table, run the lemmatizer on the test data, and write its report to a file called `lookup-output.txt`. The report has a fixed format of 22 lines which looks as follows:

```
Training statistics
Wordform types: 16879
Wordform tokens: 281057
Unambiguous types: 16465
Unambiguous tokens: 196204
Ambiguous types: 414
Ambiguous tokens: 84853
Ambiguous most common tokens: 75667
Identity tokens: 201485
Expected lookup accuracy: 0.967316238343
Expected identity accuracy: 0.716883052192
Test results
Total test items: 35430
Found in lookup table: 33849
Lookup match: 32596
Lookup mismatch: 1253
Not found in lookup table: 1581
Identity match: 1227
Identity mismatch: 354
Lookup accuracy: 0.962982658276
Identity accuracy: 0.776091081594
Overall accuracy: 0.954642957945
```

The numbers above are a correct output when running the program on the supplied data (the submission script). There is some variability possible in the output; see note below. The numbers will be different when the program is run on the data used by the grading script.

Starter code is available on Blackboard, which already reads the input and formats the output correctly. You are strongly encouraged to use this starter code; you will need to write the code that performs the counts and calculates the results correctly.

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 training and 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 `lookup-lemmatizer.py`; if your program uses auxiliary files then you must also include these in your personal directory, though for this exercise there is probably no need for auxiliary files.

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 lemmatizer 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 lemmatizer on training and test data from a different language, and compare the output of your program to a reference output for that language. Each of the 20 numbers calculated will count for 5% of the grade for the assignment.

Note

Ties in the training data. The problem definition does not state what to do in case of ties in the training data, that is when for an ambiguous word form, there are two or more most common lemmas. In this case, either of the lemmas could enter the lookup table, which could cause a small amount of variation in the test data. This is expected, and the grading script will allow for this variation; for the supplied data (the submission script), the following is legitimate (correct) variation:

- Lookup match: 32589 to 32640
- Lookup mismatch: 1209 to 1260
- Lookup accuracy: 0.962775857485 to 0.964282548967
- Overall accuracy: 0.954445385267 to 0.955884843353

Collaboration and external resources

- This is an individual assignment. You may not work in teams or collaborate 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 model parameters, as well as the use of these parameters for lemmatization, 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.