PMI Calculator

Francisco Iacobelli

October 1, 2019

1 Introduction

This assignment is to give you a better understanding of PMI and to get you to build word counts and start computing simple probabilities.

Lastly, the programming questions must be done in python and you may not use any external libraries except the standard python libraries (such as `re, collections, math`, etc.)

1.1 PMI

As a refresher, pointwise mutual information or PMI is basically a measure of how much of a contribution does a word add to another. Basically, it is the ratio of the probability of seeing the bigram vs. the probability of independently seeing the two words.

$$PMI(w_1, w_2) = log_2 \frac{Count(w_1 w_2)N}{Count(w_1)Count(w_2)}$$

Where w_1 and w_2 are two words; N is the total number of words in the vocabulary. $w_1 w_2$ is the bigram comprised of both words.

For more details, watch the videos or check the slides from the collocations class.

2 Programming Task

2.1 Tranining your PMI

In order to compute the PMI you need counts of words and bigrams. So, create a program that can read a text file from the command line and stores a file with unigrams and their frequency, bigrams and their frequency, and the total number of words in the text.

This program must be called `corpus_stats.py` and should be run like so:

`python corpus_stats.py input_file output_file` where input file is a text from which to extract the counts, and output files is the name of the file that will store the counts.

For example, if we have a text file name `cat.txt` with the following content:

```
The day we decide the future is the day we are free.
```

`python corpus_stats.py cat.txt out.dat` will produce a file named `out.dat` with the following contents:

```
the 3
day 2
we  2
decide  1
future  1
are 1
free    1
the_day 2
day_we  2
we_decide   1
decide_the  1
the_future  1
future_is   1
is_the 1
we_are  1
are_free    1
@total@ 12
```

As you can see, every words has its associated frequency next to it separated by tab. Also, bigrams are separated by _ and also have their frequency next to them. Lastly, there is some word with unlikely characters that is associated with the total number of words in the text file (N).

2.2 PMI

Create a program called `pmi.py` that takes in a counts file like the output generated earlier, two words, and produces the PMI value of those two words.

For example:`python pmi.py out.dat the future` will read `out.dat` (generated in the previous step) and output the $PMI(the, future)$. That is:

$$log_2 \frac{Count(thefuture) \times 12}{Count(the)Count(future)} = log_2 \frac{1 \times 12}{3 \times 1} = log_2 4 = 2.0$$

3 Programming Recommendations

- Make sure your code is clear and simple. If it is not, revisit it.
- The data structure you use to represent the counts and lookup words should probably be quick to lookup and should be able to be indexed by strings. The dictionary in python provides such structure.
- Practice training your PMI with a large file like `movieReviews.txt`movieReviews.txt

4 Submitting your homework

You must submit ONE zip file with the two programs. No text file. I'll use my own text files.

5 Python File Formatting

The file must comply with the following convention (THIS IS VERY IMPORTANT)

- The first line of your file should indicate the python version as follows:
  - If you are using a flavor of python 3.x, your first line should be: `#!/usr/bin/env python3`
  - If you are using a flavor of python 2.7.x your first line should be: `#!/usr/bin/env python2`
- The second line should have the name of the homework and optionally a couple of words about it. These should be enclosed in three quotation marks. For example: `""" Eliza homework. Relationship advisor """`
- The third line should have your name assigned to the `__author__` variable. For example if your name is "John Doe" your next line should be:

  `__author__="John Doe"`
- Optionally, you can specify a file encoding on your second line and then follow the convention. You do this by adding the following line as a second line: `# # -*- coding: utf-8 -*-`

A sample hello world file created with python 2.7.x for John Doe looks like this:

```
#!/usr/bin/env python2
# # -*- coding: utf-8 -*-
""" Hello World program """
__author__="John Doe"

print "Hello World"
```

The same file created with python 3.x looks like:

```
#!/usr/bin/env python3
# # -*- coding: utf-8 -*-
""" Hello World program """
__author__="John Doe"

print ("Hello World")
```