UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG
School of Computer Science & Applied Mathematics

Artificial Intelligence (COMS4033A/7044A)
Bayesian Topic Modelling

1 Introduction

In this lab, we'll be building a Bayesian model to determine whether a piece of text belongs to a particular subject. In this case, we will build a system that can predict, given a piece of text, which of ten Shakespeare plays it most likely belongs to. Since AI is about picking the best tool for the job, this lab will be conducted in *Python*.

In the submissions that follow, we will work our way to constructing a naïve Bayes classifier. This will require us to process a bunch of text and build a bag-of-words model from which we can compute the necessary statistics. Recall that the formula for Bayesian topic modelling (where the random variable for a topic S here equates to different plays) is as follows:

$$P(S|W_1,\ldots,W_n) = \frac{P(W_1,\ldots,W_n|S)P(S)}{P(W_1,\ldots,W_n)}; \quad P(W_1,\ldots,W_n|S) = \prod_i P(W_i|S)$$

We will therefore need to compute a count for each word in each play, which also involves extracting the words from the play's text. For all of the submissions that follow, you should use the ten provided text files that contain ten of Shakespeare's plays. These are:

1. *The Merchant of Venice* (`merchant.txt`)
2. *Romeo and Juliet* (`romeo.txt`)
3. *The Tempest* (`tempest.txt`)
4. *Twelfth Night* (`twelfth.txt`)
5. *Othello* (`othello.txt`)
6. *King Lear* (`lear.txt`)
7. *Much Ado About Nothing* (`ado.txt`)
8. *Midsummer Night's Dream* (`midsummer.txt`)
9. *Macbeth* (`macbeth.txt`)
10. *Hamlet* (`hamlet.txt`)

Submission 1: Word Cleaning (10 marks)

Before we can build our model, we need to extract words from a thousands of lines of text. These words will be separated by spaces and blank lines, and will also contain punctuation which should be removed. Write a Python program that accepts a string on a single line and outputs the words as they appear in order in lowercase. Words are pieces of text that are separated by spaces (so use the Python `split()` function) but should have all non-alphabetical characters removed

Input

Input is a single line of text, potentially containing characters, whitespace (both leading and trailing), digits and punctuation.

Output

Print the words as they appear in order in lowercase with only letters appearing. Each word should be separated by a single blank space.

Example Input-Output Pairs

Sample Input #1

```
Your Grace hath ta'en great pains to qualify
```

Sample Output #1

```
your grace hath taen great pains to qualify
```

Sample Input #2

```
_Juliet._ How art thou out of breath, when thou hast breath?
```

Sample Output #2

```
juliet how art thou out of breath when thou hast breath
```

Sample Input #3

```
To hear good counsel; O, what learning is!--                    160
```

Sample Output #3

```
to hear good counsel o what learning is
```

Submission 2: Bag-of-words (30 marks)

Now that we can extract words from a line, we need to extract all the words from a play and build a frequency count of how often each word appears. To do this, your Python program will need to read from a text file. Download all ten plays from Moodle and save them in the same directory as your program. To read in each line of a given play from a file called `filename`, you should use the following code

```
with open(filename, 'r', encoding='utf-8') as file:
    for line in file:
        # do something with line
```

Write a program that accepts a filename, reads in the entire file and counts the number of times each word (where word is defined in the previous submission). Your program should compute these statistics and output the top three most commonly found words.

Input

Input is a single line of text, indicating the file to read from.

Output

Print a single line containing three words separated by a space. The first word is the most common word in the file, the second is the second most common and the third is the third most common.

Example Input-Output Pairs

Sample Input #1

```
merchant.txt
```

Sample Output #1

```
the i and
```

Sample Input #2

```
romeo.txt
```

Sample Output #2

```
and the i
```

Sample Input #3

```
twelfth.txt
```

Sample Output #3

```
i the and
```

Submission 3: Classifying sentences (50 marks)

Now that we can compute the bag-of-words model for each play, we can build a classifier to determine which play is more likely to contain a given sentence. In this submission, we are only interested in determining the most likely play, not computing the exact probabilities. This means we can ignore the denominator in the naïve Bayes equation, since all calculations will have that in common. To determine if play a is more likely than play b, we need to check whether:

$$P(W_1, \ldots, W_n | S = a)P(S = a) > P(W_1, \ldots, W_n | S = b)P(S = b)$$

However, this formula is problematic — if each of the probabilities is very small, and there are many words, then we risk underflowing. To prevent numerical instability, we should therefore compute the log of the numerator. This still holds because if the above holds, then so does

$$\log P(W_1, \ldots, W_n | S = a)P(S = a) > \log P(W_1, \ldots, W_n | S = b)P(S = b)$$

There are ten topics, represented by each of the plays. Your program should read in each textfile and compute the bag-of-words model for each. Then using the formula above, determine given a new sentence which play it is most likely from. For this case, you should use a uniform prior (i.e. each play is equally likely under the prior).

Input

Input consists of a single line. The line should be processed exactly like the previous submissions to extract words.

Output

Output the most likely play. Use the exact names of the plays as they appear on the first page of this document.

Example Input-Output Pairs

Sample Input #1

```
maintained a politic state!
```

Sample Output #1

```
Much Ado About Nothing
```

Sample Input #2

```
Men, heaven and devils confess'd
```

Sample Output #2

```
Othello
```

Sample Input #3

```
__His mother was a strong witch!__
```

Sample Output #3

```
Macbeth
```

Submission 4: Full Estimates (10 marks)

The previous example showed how to find the most likely play (also known as the *maximum a posteriori*). However, it would be nice if we could associate probabilities with each of the plays. This will require us to compute the full posterior, instead of just the numerator term. Again, we will assume a uniform prior

However, we're going to run into even more problems with numerical issues, since now we have the denominator to worry about to. To overcome this, we can use a similar trick with logarithms to avoid this. That is,

$$\log\left[\frac{P(W_1,\ldots,W_n|S)P(S)}{P(W_1,\ldots,W_n)}\right] = \log P(W_1,\ldots,W_n|S)P(S) - \log P(W_1,\ldots,W_n)$$

The numerator is calculated in the same manner as previously. For the log of the denominator, for each play/class C_k, we can rewrite this as

$$\log\left(\sum_{k=1}^{|C|} P(\mathbf{W}|S=C_k)P(S=C_k)\right) = \log\left(\sum_{k=1}^{|C|} \exp\left(\log\left(P(\mathbf{W}|S=C_k)p(S=C_k)\right)\right)\right)$$

The denominator now has the form $\log\sum_k e^{a_k}$, which means we can use the **log-sum-exp** trick:

$$\log\sum_k e^{a_k} = \log\sum_k e^{a_k}e^{A-A} = A + \log\sum_k e^{a_k-A}$$

where

- $a_k = \log\left(P(\mathbf{W}|S=C_k)P(S=C_k)\right)$,
- $A = \max_{k\in\{1,\ldots,|C|\}} a_k$.

Having computed the numerator and denominator, we simply subtract one from the other and exponentiate the result to produce the probability of the play being true given the sentence.

Input

Input consists of a single line. The line should be processed exactly like the previous submissions to extract words.

Output

Output the probabilities associated with each play, ordered from most likely to least (you can break ties however you like). Probabilities should be multiplied by 100 and rounded to the nearest whole number. Your output format should look something like this:

```
Macbeth: 88%
Hamlet: 6%
The Tempest: 4%
Much Ado About Nothing: 2%
King Lear: 0%
Romeo and Juliet: 0%
The Merchant of Venice: 0%
Midsummer Night's Dream: 0%
Othello: 0%
Twelfth Night: 0%
```

Note: The numbers above are invented. This is just to show you the format of the output

Submission

Submit your code and a text file containing the output when your code is run with the following input:

```
You shall be king.
```

Your output and code will be manually marked.