Assignment 3: Bayesian Inference

EECS 492: Artificial Intelligence

Fall 2015

Now: Thursday, October 29, 9:00 am

Due: Tuesday, November 17, 11:00 pm

In this assignment, you will implement one flavor of Naive Bayes (NB) classifier – the Bernoulli Naive Bayes – to solve an interesting problem in Natural Language Processing (NLP). Naive Bayes classifiers very succinctly and beautifully illustrate the core principle of Bayesian Inference – gathering evidence from many small sources, and accumulating them using a probabilistic framework to yield a single large signal. You will use the classifier to solve Authorship Attribution – a long-standing problem in NLP.

We will provide you with the data (text documents) and tokenizer,[1] and you will write your own code to implement the classifier (both training and testing), extract features, and produce output in the submission format (explained below). The Naive Bayes classifier is explained below.

Let d be a document, c be a class label, C is the set of all class labels, and $P(c|d)$ is the conditional probability of the class label given the document. We can now define the best class label c^* for d as the one that maximizes $P(c|d)$:

$$c^* = argmax_{c \in C} \left[P(c|d) \right] \tag{1}$$

We will call this a *maximum a posteriori (MAP)* classifier because we will use Bayes Rule to compute $P(c|d)$ as a posterior probability from the prior probability $P(c)$ and the likelihood $P(d|c)$.

By Bayes Rule, we have:

[1] The data is available here: `http://www.mathcs.duq.edu/~juola/problems/problemset.tar.gz`. Please feel free to download it and get started.

$$P(c|d) \propto P(c)P(d|c) \tag{2}$$

and $P(d|c)$ is approximated as:

$$P(d|c) \approx \prod_{i=1}^{n} P(f_i|c) \tag{3}$$

where document d is *represented by* n features f_1, f_2, $\ldots f_n$. Each feature imparts a small amount of evidence into the model. Note that we assume f_i's are *independent* given the class label c. This is called the *conditional independence assumption* of Naive Bayes classifiers, and is the reason why they are called "Naive". Despite this assumption, however, Naive Bayes works surprisingly well in practice.

We usually estimate $P(c)$ and $P(f_i|c)$ on the training data. The estimated value for $P(c)$ is $\hat{P}(c)$, and the estimated value for $P(f_i|c)$ is $\hat{P}(f_i|c)$. The final form of the classifier looks as follows:

$$c^* = argmax_{c \in C} \ \left[\hat{P}(c) \prod_{i=1}^{n} \hat{P}(f_i|c)\right] \tag{4}$$

Since probabilities are small, and multiplication makes them smaller, a straight-forward implementation of Equation (4) is very likely to underflow, and return a value of zero. However, we are actually interested in the value c^* of c that maximizes the expression, rather than the actual maximum value. Since log is a monotonic function, we can take the log of Equation (4) and get the same maximum in Equation (5) without the danger of underflow.[2]

$$c^* = argmax_{c \in C} \ \left[log \ \hat{P}(c) + \sum_{i=1}^{n} log \ \hat{P}(f_i|c)\right] \tag{5}$$

One of the principal variants of Naive Bayes classifier is the Bernoulli Naive Bayes, where $\hat{P}(f_i|c)$ is computed as:

$$\hat{P}(f_i|c) = \frac{N_{ci} + 1}{N_c + 2} \tag{6}$$

where N_{ci} is the number of documents with class label c that contain feature f_i, and N_c is the number

[2]Please use log base 2 in all cases.

of documents with class label c.

A natural generalization that is useful for other problems is the Multinomial Naive Bayes classifier, with the following more general formula for $\hat{P}(f_i|c)$:

$$\hat{P}(f_i|c) = \frac{T_{ci} + 1}{\sum\limits_{j=1}^{|V|} T_{cj} + |V|} \tag{7}$$

where T_{ci} is the number of times feature f_i appeared under class label c, and V is the set of unique features in the training data (also known as the *vocabulary*). The addition of one in the numerator and $|V|$ in the denominator is called *add-one smoothing*. It is done so that zero probabilities of unseen features cannot make the whole likelihood zero.

For both Bernoulli and Multinomial Naive Bayes, prior probability estimate $\hat{P}(c)$ is computed as follows:

$$\hat{P}(c) = \frac{N_c}{N} \tag{8}$$

where N is the total number of documents (in the training set).

Naive Bayes classifier has been briefly touched upon in AIMA pages 499 and 808. For more details on the two flavors of Naive Bayes, please see Chapter 13 of the book "Introduction to Information Retrieval" by Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze (available on CTools under Resources/readings/13bayes.pdf).

Authorship Attribution

Now that we know about the Naive Bayes classifier, the next step is to use it to solve a practical problem in NLP. In Authorship Attribution, you are given documents with known authors. Your task is to train a model on these documents, and then test the model on documents of unknown authorship. For example, say we would like to know if a particular play of Shakespeare was written by Shakespeare or Marlowe. One way to answer that question will be to train a classifier (e.g., Naive Bayes) on the plays of Shakespeare and Marlowe, and then use the trained model on the disputed play to see what our model predicts. The success of computational stylometry in this regard stems partly from the fact that Bayesian reasoning can be very effective at teasing apart the *authorial fingerprints* of individual people. It appears that stop words[3] are good candidate features for this problem, since authors have little conscious control over the choice of stop words. Several lists of English stop words are available online. We have provided you with the one available at `http://www.lextek.com/manuals/onix/stopwords1.html`. This is a standard list, and has been used in several research papers.

The dataset we will use for this assignment, is a subset of the Ad-hoc Authorship Attribution Competition (AAAC) Dataset, publicly available at `http://www.mathcs.duq.edu/~juola/authorship_materials2.html`.[4] The dataset is divided into 13 problems, the first eight being in English, and the rest in four other languages (French, Latin, Dutch, and Serbian-Slavonic). We will use Problems A, B, C, G, and H for this assignment. Each problem has a separate directory: "problemA", "problemB", "problemC", and so on. Within a particular directory, say "problemA", we have training examples with known authors:

- Atrain01-1.txt (example 1 from author 01)
- Atrain01-2.txt (example 2 from author 01)
- Atrain01-3.txt (example 3 from author 01)
- ...
- Atrain13-1.txt (example 1 from author 13)
- Atrain13-2.txt (example 2 from author 13)
- Atrain13-3.txt (example 3 from author 13)

and test examples with unknown authors:

- Asample01.txt
- Asample02.txt
- Asample03.txt
- and so on.

"problemA" directory has 13 test examples.

[3]Very frequently used English words, such as "a", "an", "the", "am", "is", "are", etc. Stop words are also often called *function words*, as they carry very little semantic content.

[4]Please make sure to download the complete dataset as a tarball: `http://www.mathcs.duq.edu/~juola/problems/problemset.tar.gz`.

For the assignment, we have provided you with a ground truth file ("test_ground_truth.txt") that contains author information for the test examples of each AAAC problem. You will use this file to obtain the test accuracy of your classifier.

Assignment

Please implement the Bernoulli Naive Bayes classifier, as described above, for the Authorship Attribution problem. Tokenize the text using the given code.[5] For Authorship Attribution, please use the given stop words as features. Performance of your classifier will be evaluated on the AAAC test examples. The performance measure we will use, is called *accuracy.* It is the percentage of correctly classified test examples. Please make sure your code takes one argument – the AAAC directory containing training and test documents for a particular problem. You may assume that the stop words file and ground truth file ("test_ground_truth.txt") are present in the same directory as your code.

Output the following for each problem (A, B, C, G, and H):

1. **Test Accuracy**: What is the accuracy on the test data?
2. **Confusion Matrix**: What is the confusion matrix?[6]
3. **Feature Ranking**: Rank top 20 features by their *class-conditional entropy* (on training data). The class-conditional entropy (CCE) for a particular feature f_i is estimated as:

$$CCE_i \approx -\sum_{c \in C} \left[\hat{P}(c)\ \hat{P}(f_i|c)\ (log\ \hat{P}(f_i|c))\right]^7 \tag{9}$$

   For each feature, compute its class-conditional entropy. Then sort the features in descending order of their entropy, and present the top 20 features along with their entropy values.

4. **Feature Curve**: Sort the features in the descending order of their frequency (in the training data). Then take top 10 features, and train your model using a vocabulary of these top 10 features. Save the test accuracy of this model. Then take top 20 features, and save the test accuracy of the corresponding model. Then take top 30, top 40, and so on, and for each – save the corresponding test accuracy. In the end, plot the test accuracy against number of features.[8] What conclusion can be drawn from this curve?

[5]Tokenization involves lowercasing the text, removing all punctuation, and splitting the text into words.

[6]It is a matrix where the rows are "ground truth" class labels, and columns are predicted classes. An example is given here: `https://en.wikipedia.org/wiki/Confusion_matrix`. We have provided you with a function to print out the confusion matrix.

[7]Please use log base 2 in all cases.

[8]X-axis is number of features, Y-axis is test accuracy.

What to submit

A zipped directory with one subdirectory: **code**, and one PDF file. Name your directory "A3-uniquename", where "uniquename" is replaced by your own uniquename. The **code** subdirectory contains the code, the given stop words file, the given ground truth file, and a README with enough information to run your code. Specifically, your program should take one argument: the AAAC directory containing training and test files, and it should print out the test accuracy, the confusion matrix, the feature ranking, and the test accuracy for top k features (k varying from 10 to the maximum in steps of 10). We will go through the code to make sure they are present. Please submit on CTools, through the Assignments tool. If for any reason that doesn't work, submit it to the Drop Box, and send email.

Here are two sample commands for C++ and python:[9]

```
./mycode AAAC_problems/problemA/
```

```
python mycode.py AAAC_problems/problemA/
```

Now, please answer the following questions (they are graded).[10] Once you are done, save this PDF as "answers.pdf" under "A3-uniquename" directory, zip it, and upload on CTools.

AAAC Problem A

- What is the test accuracy?[11]

  The test accuracy was 0.384615384615

- What is the confusion matrix?

  The confusion matrix was

0	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	0	0	0	0	0	0	0	1	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	1	0	0
3	0	0	0	0	0	1	0	0	0	0	0	0	0
4	0	0	0	1	0	0	0	0	0	0	0	0	0
5	0	0	0	0	1	0	0	0	0	0	0	0	0
6	1	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	1	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	1	0	0
9	0	0	0	0	0	0	0	0	1	0	0	0	0
10	0	0	0	0	0	0	0	0	0	1	0	0	0
11	0	0	0	0	1	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	0	0	0	0	0	0	0	1	0	0	0	0

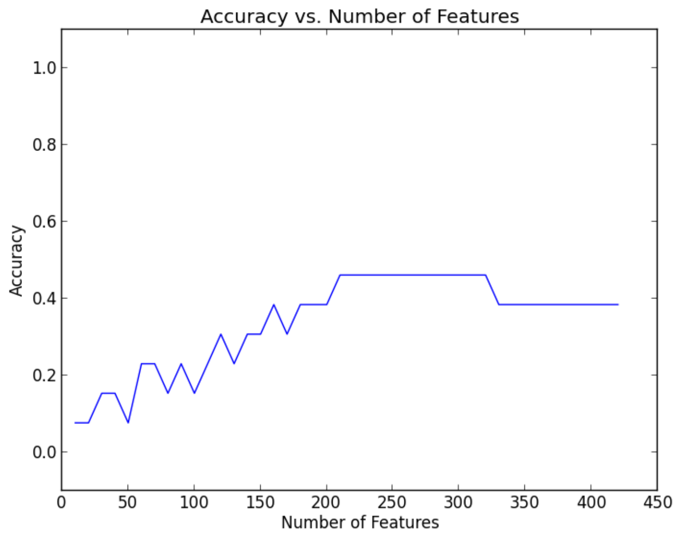
[9] For C++, you may want to use something like the following command: `g++ -O3 -std=c++11 mycode.cpp -o mycode`.
[10] Please fill in the PDF directly. To edit PDF, you can use an online utility like `https://www.pdfescape.com/`.
[11] We got 38.4615% with our implementation.

- What is the feature ranking? Please list the top 20 features, along with their class-conditional entropy.

The feature ranking was

Feature	Entropy
going	0.512007736726
against	0.510254610741
everyone	0.510254610741
under	0.506924661543
find	0.506748358771
always	0.503418409573
need	0.503418409573
greater	0.501841586361
having	0.501841586361
others	0.501665283588
get	0.500088460376
thought	0.500088460376
until	0.500088460376
areas	0.496758511179
off	0.496758511179
working	0.496758511179
high	0.496758511179
men	0.496758511179
two	0.496582208406
give	0.495678014905

- Please give the feature curve as described above (should be legible). What conclusion can be drawn from this curve?



From this curve we can conclude that after around 300 features, the program begins to overfit the data

AAAC Problem B

- What is the test accuracy?

  The test accuracy was 0.307692307692

- What is the confusion matrix?

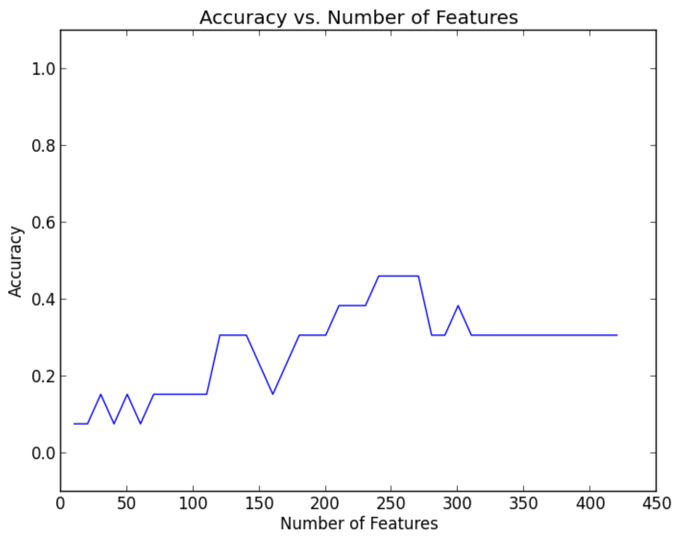
```
The confusion matrix was
0 1 2 3 4 5 6 7 8 9 10 11 12 13
1 0 0 0 0 0 0 0 0 0 1 0 0 0 0
2 0 0 0 0 0 0 0 0 0 0 1 0 0 0
3 0 0 0 0 0 0 0 0 0 0 1 0 0 0
4 0 0 1 0 0 0 0 0 0 0 0 0 0 0
5 0 0 0 0 0 0 0 0 0 0 1 0 0 0
6 0 0 0 0 0 1 0 0 0 0 0 0 0 0
7 0 0 0 0 0 0 1 0 0 0 0 0 0 0
8 0 0 0 0 0 0 0 0 0 0 0 1 0 0
9 0 0 0 0 0 0 0 0 0 0 1 0 0 0
10 0 0 0 0 0 0 0 0 0 0 1 0 0 0
11 0 0 0 0 0 0 0 0 0 1 0 0 0 0
12 0 0 1 0 0 0 0 0 0 0 0 0 0 0
13 0 0 0 0 0 0 0 0 0 0 0 0 0 1
```

- What is the feature ranking? Please list the top 20 features, along with their class-conditional entropy.

  The feature ranking was

```
Feature    |    Entropy
----------------------------
going 0.512007736726
against 0.510254610741
everyone 0.510254610741
under 0.506924661543
find 0.506748358771
always 0.503418409573
need 0.503418409573
greater 0.501841586361
having 0.501841586361
others 0.501665283588
get 0.500088460376
thought 0.500088460376
until 0.500088460376
areas 0.496758511179
off 0.496758511179
working 0.496758511179
high 0.496758511179
men 0.496758511179
two 0.496582208406
give 0.495678014905
```

- Please give the feature curve as described above (should be legible). What conclusion can be drawn from this curve?



From this curve, we conclude that after around 250 features, the program begins to overfit the data.

AAAC Problem C

- What is the test accuracy?

  The test accuracy was 0.444444444444

- What is the confusion matrix?

The confusion matrix was

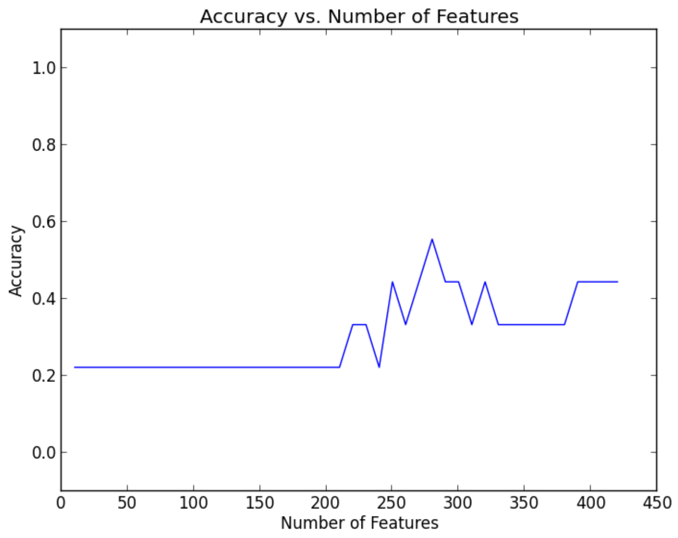
```
0 1 2 3 4 5
1 0 2 0 0 0
2 0 2 0 0 0
3 0 1 0 0 0
4 0 0 0 2 0
5 2 0 0 0 0
```

- What is the feature ranking? Please list the top 20 features, along with their class-conditional entropy.

The feature ranking was

Feature	Entropy
differently	0.525068453926
yours	0.525068453926
gets	0.51840472838
groups	0.51840472838
nowhere	0.513706285872
cases	0.511741002833
showing	0.511741002833
ends	0.507042560325
working	0.507042560325
f	0.502128748016
h	0.502128748016
puts	0.495465022469
wanting	0.495465022469
finds	0.492516493199
backs	0.490766579961
downs	0.490766579961
goods	0.490766579961
m	0.490766579961
mostly	0.490766579961
d	0.486847827759

- Please give the feature curve as described above (should be legible). What conclusion can be drawn from this curve?



From the curve, we conclude that after around 270 features, the program begins to overfit the data

AAAC Problem G

- What is the test accuracy?

  The accuracy was 0.25

- What is the confusion matrix?

  The confusion matrix was

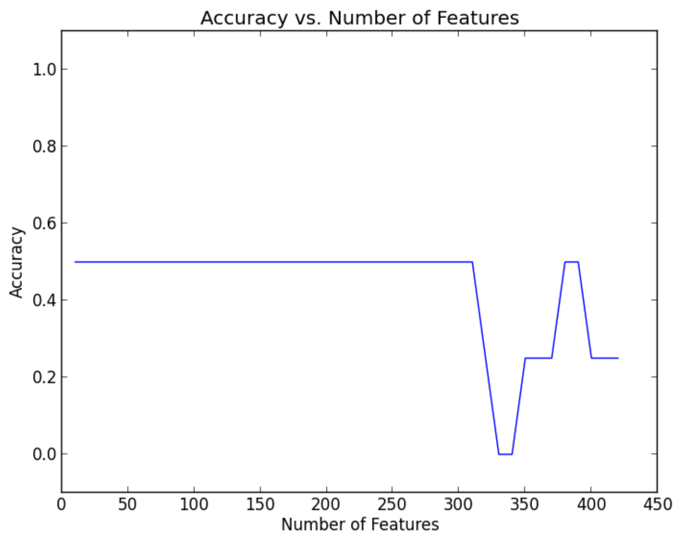
```
0 1 2
1 1 1
2 2 0
```

- What is the feature ranking? Please list the top 20 features, along with their class-conditional entropy.

  The feature ranking was

```
Feature   |   Entropy
everyone 0.528771237955
keeps 0.528771237955
problems 0.528771237955
somebody 0.528771237955
wanting 0.528771237955
c 0.496578428466
cases 0.496578428466
differently 0.496578428466
evenly 0.496578428466
everybody 0.496578428466
f 0.496578428466
furthered 0.496578428466
g 0.496578428466
generally 0.496578428466
grouped 0.496578428466
k 0.496578428466
m 0.496578428466
mr 0.496578428466
mrs 0.496578428466
nobody 0.496578428466
```

- Please give the feature curve as described above (should be legible). What conclusion can be drawn from this curve?



From the curve, we conclude that after around 300 features, the program begins to overfit the data

AAAC Problem H

- What is the test accuracy?

The test accuracy was 0.666666666667

- What is the confusion matrix?

The confusion matrix was

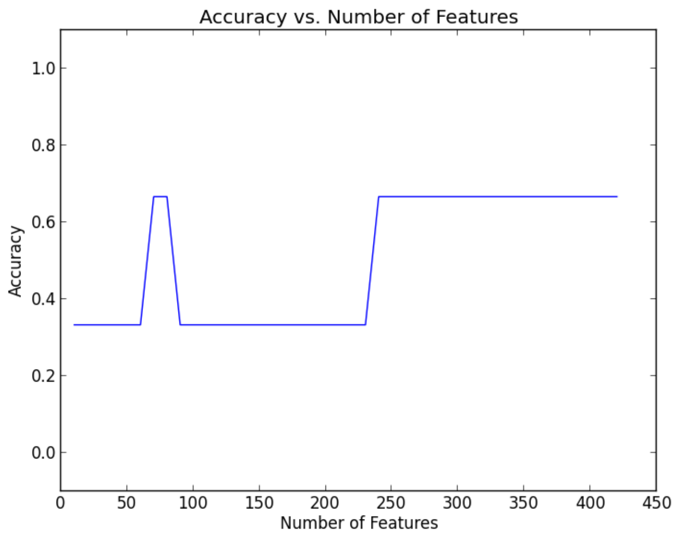
```
0 1 2 3
1 1 0 0
2 0 1 0
3 0 1 0
```

- What is the feature ranking? Please list the top 20 features, along with their class-conditional entropy.

The feature ranking was

Feature	Entropy
above	0.528320833574
across	0.528320833574
although	0.528320833574
anywhere	0.528320833574
asking	0.528320833574
asks	0.528320833574
b	0.528320833574
backed	0.528320833574
backing	0.528320833574
backs	0.528320833574
became	0.528320833574
becomes	0.528320833574
began	0.528320833574
behind	0.528320833574
beings	0.528320833574
big	0.528320833574
c	0.528320833574
certain	0.528320833574
clear	0.528320833574
clearly	0.528320833574

- Please give the feature curve as described above (should be legible). What conclusion can be drawn from this curve?



From the feature curve, it looks like the program still fits the data well after 400 features, but it usually under fits the data for less than 250 features.

Scoring Details

Please make sure your code runs on CAEN before you submit.

We will grade the following on both completion and correctness. That is, each part has to be complete **and** correct to receive full credit.

- Code compiles and runs on CAEN (10 points)
- Test accuracy (5 problems * 4 points each = 20 points)
- Confusion matrix (5 problems * 4 points each = 20 points)
- Feature ranking (5 problems * 4 points each = 20 points)
- Feature curve (5 problems * 4 points each = 20 points)
- Code prints the values for feature curve[12] (5 problems * 2 points each = 10 points)

Total: 100 pts.

Extra Credit

Please fill in the PDF directly. To edit PDF, you can use an online utility like `https://www.pdfescape.com/`.

- Implement the Multinomial Naive Bayes classifier to solve another interesting problem in NLP – Language Identification. In Language Identification, the goal is to identify the language of a given document. Train a classifier on documents with known language, and test it on other documents. Language identification is interesting because people often mix two or more languages while writing, esp. on social media like Facebook and Twitter. This phenomenon is variously known as *code switching* or *code mixing*. Given that we can automatically identify people's language, it becomes easier to analyze cultural, social, political, and economic backgrounds of a large section of human population who do not necessarily speak English. For language identification problems, character bigrams (sequences of two characters) have been found to perform well.[13] What dataset to use for Language Identification?

[12]Test accuracy for top k features, k varying from 10 to the maximum in steps of 10.

[13]Example of character bigrams: the phrase "hot dog." has seven character bigrams – "ho", "ot", "t ", " d", "do", "og", and "g.". Space characters and punctuation symbols are considered valid while extracting character bigrams.

A good choice will be a subset of the Europarl dataset.[14] You may use character bigrams (with punctuation and spaces) as features. What is the test accuracy, confusion matrix, feature ranking, and feature curve for Language Identification?

- Instead of character bigrams, use character trigrams (three-character sequences) for Language Identification. Does it make any difference? How about higher-order character n-grams?

- Do word n-grams have an impact on performance ($n > 1$)?[15]

[14] `http://www.statmt.org/europarl/`.

[15] The sentence "I go to UMich" has three word bigrams – "I go", "go to", and "to UMich", and two word trigrams – "I go to", and "go to UMich".

- Instead of feature frequency, use a different measure of salience, e.g., *tfidf*.[16] What difference does it make?

- *Stemming* refers to the practice of partial or complete removal of inflection from a word. For example, "run", "runs", and "running" are all stemmed to the base form "run". Several English stemmers are available online. Please use the Porter Stemmer. Does stemming have any impact on performance of Authorship Attribution and/or Language Identification?

- Does the *length* of the text samples in number of words/characters (for training and/or test) have an impact on performance?

[16] *tfidf*, or *term frequency inverse document frequency*, is defined as the product of term frequency (tf) and inverse document frequency (idf). The formula to compute *tfidf* is explained here: `https://en.wikipedia.org/wiki/Tf%E2%80%93idf`. Please note that the term frequency component needs to be normalized to prevent a blow-up of score due to high-frequency terms.