**CS 441 Artificial Intelligence**
**Assignment 4 (Deadline January 5th, 2018)**

In this assignment, you will implement the Naive Bayes classification method and use it for sentiment classification of customer reviews.

Write Python code to solve the tasks described below. There are also some discussion questions (colored blue). Send the code and the answers by email (answers should be in a pdf) to Serhan (serhan.aksoy@antalya.edu.tr).

`all_sentiment_shuffled.txt` contains a corpus of customer reviews from six of the review topics used in the paper by Blitzer et al., (2007). The data has been formatted so that there is one review per line, and the texts have been tokenized and normalized. Here is an example of a line.

```
music neg 544.txt i was misled and thought i was buying the entire cd and it contains one song
```

A line in the file is organized in columns:

- 0: topic category label (books, camera, dvd, health, music, or software)
- 1: sentiment category label (pos or neg)
- 2: document identifier
- 3 and on: the document tokens

Here is some Python code to read the entire corpus. This code is also available in the file assignment4.py in the package you unpacked.

```
def read_corpus(corpus_file):
    out = []
    with open(corpus_file) as f:
        for line in f:
            tokens = line.strip().split()
            out.append( (tokens[1], tokens[3:]) )
    return out
```

We first take away the topic label (`tokens[0]`), which you don't need unless you solve the last optional task.

In the code below, we read all the data and split the data into a training and an evaluation part. For instance, we may use 80% for training and the remainder for evaluation.

```
all_docs = read_corpus('all_sentiment_shuffled.txt')
split_point = int(0.8*len(all_docs))
train_docs = all_docs[:split_point]
eval_docs = all_docs[split_point:]
```

**Training the Naïve Bayes classifier and classifying new documents**
Write a Python function that uses a training set of documents to estimate the probabilities in the Naive Bayes model. Return some data structure containing the probabilities. The input parameter of this function should be a list of documents with sentiment labels, i.e. a list of pairs like train_docs above. It could look something like this:

```
def train_nb (training documents):
     ...
    (return the data you need to classify new instances)
```

**Note: When you are calculating the feature probabilities you must apply Laplace smoothing. Try different values and discuss the results you get with different smoothing parameters.**

Then write a Python function that classifies a new document. The inputs are 1) the probabilities returned by the first function; 2) the document to classify, which is a list of tokens.

```
def classify_nb(classifier_data, document):
      ....
     return result
```

**`result`** should be a tuple where the first element is the predicted class label, and the second element is the probability estimate for that class label. Note the function above should return normalized probability i.e., the sum of the probability estimate for pos and neg should sum up to 1.

Also, in this function you need to think about numeric problems. If you multiply many small probabilities you may run into problems with numeric precision: the probability becomes zero. To handle this problem, I recommend that you compute the *logarithms* of the probabilities instead of the probabilities. To compute the logarithm in Python, use the function log in the math library. Note that log(P1 * P2) = log(P1) + log(P2), so if you use log probabilities you will sum them instead of multiplying. (check slide 22 of Lecture21.pdf)

### Evaluating the classifier

We will evaluate the classifier carefully in the second assignment. In this assignment, we just compute the *accuracy*, i.e. the number of correctly classified documents divided by the total number of documents. Write a function that classifies each document in the test set, compares each label to the gold-standard label, and returns the accuracy.

```
def evaluate_nb(classifier_data, evaluation_documents):
    ...
    (return the accuracy)
```

What accuracy do you get when evaluating the classifier on the test set?

## Error analysis

Find a few misclassified documents and comment on why you think they were hard to classify.

### Bonus [30 pts]

Try to think of ways to improve the accuracy by changing the features used for classification.

One possible improvement could be to try to do something about negation, since it is obvious that negation is not well handled by the bag-of-words approach we are using. Pang et al., (2002) used this trick: they added a special negation marker to all tokens appearing in a sentence after a negation (*not*, *n't* etc).

Many people use *bigram* features in addition to single-word features. Can you improve your classifier by using bigrams?

**Credit.**
**The assignment is based on an assignment given in STATNLP class at University of Gothenburg.**

### References


- **Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan**: *Thumbs up? Sentiment Classification using Machine Learning Techniques*. In Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002).
- **John Blitzer, Mark Dredze, and Fernando Pereira**: *Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification.* In Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL 2007).