LINKÖPING UNIVERSITY
Department of Computer and Information Science
Division of Statistics and Machine Learning
Måns Magnusson/Mattias Villani

2015-11-27
Text Mining
732A47

Statistical Models for Textual Data - Computer Lab

Deadline:	See LISAM
Teacher:	Måns Magnusson
Grades:	Pass/Fail
Submission:	Through LISAM

Write your solutions in COMPLETE, READABLE and EXECUTABLE code.
Solutions should be written in a text file with .py extension.
Graphs produced during the lab should NOT be submitted, submit only the code.
Comment directly in the code (using # or ”’) whenever something needs to be explained.

Note: you may need to do download the movie_reviews corpus from nltk by (see the nltk book for details)

```
import nltk
nltk.download()
```

Choose d for download and then type `movie_reviews`.

1. Document classification

   (a) Use the `movie_reviews` data from the `nltk.corpus` module to construct a naive Bayes classifier from a training sample consisting of 80% of the data. Use only binary features [`has('word')`] based on the 1000 most frequent words in the corpus. Do not pre-process the text in the reviews. Evaluate the predictive performance on the test data (the 20% that you left out from training) using measures like *Accuracy*, *Precision*, *Recall* and the *F-measure.*
   [Hint 1: To get a head start, check out the `movie_reviews` example in the NLTK book in Chapter 6. http://nltk.googlecode.com/svn/trunk/doc/book/ch06.html. Make sure that you understand the code in the NLTK `movie_review` example. You will need to modify the code later on, so it will pay off to understand it.]
   [Hint 2: Use the shuffle function in the random module to randomly reorder the documents in the collection (otherwise you might get all the positive reviews in the test set, or vice versa)]
   [Hint 3: When you have trained your naive Bayes classifier you should have a object of the type `NaiveBayesClassifier` in working memory. This object has a lot of useful methods, type the objects name followed by a period to see a list of its methods. Use Google to learn about these methods.]

   (b) Redo the analysis in a), but this time do a careful pre-processing of the data before training and prediction. Play around with normalization, stemming, removing stop words, removing punctuation, remove numbers, or whatever you find interesting. Use the same subset for training and testing as in a). Compare the predictive performance of the models to the model in a). Be creative. Analyze.

(c) Continue with the best version of the pre-processed data from b), but now do a more careful selection of features. Consider not only binary features, but also perhaps frequencies of individual words, measures of lexical diversity, average word length, maybe bigrams or collocations. Note that you need to convert the non-binary features to a set of binary features ($x < c_1$, $c_1 \leq x < c_2$ and so on, where c_1 and c_2 etc are cut-off points for the bins, see my slides from Lecture 2). Note also that k bins can be represented by $k-1$ binary features. Experiment as much as you like and have the time for. Always compare the predictive performance of the models.

(d) Finally, consider the use of tf–idf weighted features.

Have fun!