COMP3608 Artificial Intelligence
Semester 1, 2012

Assignment 2. Email Classification

Submission deadline: Monday, week 12, 12noon (28 May).
Late policy submission: A penalty of -1 mark will apply for each day late and the assignment will not be accepted if it is submitted more than 7 days after the due date. The cut-off time is 12noon.
This assignment can be completed individually or in pairs. Working in pairs is encouraged. Both students will receive the same mark.
Submission instructions: You need to submit:

1) Hard copy (report + code + plagiarism cover sheet) in the locker COMP3308/3608 (School of IT building, level 1, in the undergraduate labs wing, close to the room where the AI tutorials are held) and
2) Electronic copy (report + code + the two data files: `subject.csv` and `body.csv`) via eLearning. All files should be zipped together in a single file. The zip file should be named 0123456.zip, where 0123456 is your SID. In case of a pair submission, put both SIDs separated by an underscore: 0123456_0789123.zip. Only one of the two students needs to submit.

Programming language: You can write the program in a language of your choice (e.g. Java, C, C++, Python, Matlab) but we need to be able to test your code on the University machines. You need to include instructions how to run your code.
Weight: This assignment is worth 20 marks = 20% of your final mark

The goal of this assignment is to:
1) Learn about text classification
2) Implement the Naïve Bayes algorithm and the stratified cross validation method
3) Evaluate the classification performance of the implemented Naïve Bayes and other classifiers from Weka on a real dataset (spam email classification). Naïve Bayes classifiers are widely used in spam filters; they are simple but work surprisingly well.
4) (Extension) Implement your own extension to try to improve the accuracy of the spam filter.

1. Learn about text classification

Text classification is the task of assigning classes to text documents [1]. For example, we can classify emails (i.e. text documents) into 2 classes: spam and non-spam. We can do this automatically using machine learning algorithms. To do this we need to suitably represent each document.

One popular representation is called *bag-of-words*. It treats each word as a feature. Given a collection of documents, all words appearing in the collection are first extracted (these are the features). A *feature selection* is performed to select a subset of these words, i.e. the most representative and discriminative words based on a given heuristic. For example, one of the most popular feature selection methods is called *Document Frequency (DF)*. It computes the DF for every word in the collection (i.e. the number of documents in which the word occurs) and then selects the *N* words with highest DF. Only these words are used to build a classifier for the text documents. The assumption behind DF is that the more frequent words are more representative of the content of the document than the less frequent ones. Other popular feature selection methods are Information Gain, Mutual Information, Odds Ratio, χ^2; Categorical Proportional Difference is a newer and simple method, shown to work well [2].

Each document is then represented in terms of these selected words with a numerical value called weighting (to facilitate machine learning). The most popular weighting method is called tf-idf [1]:

$$tf-idf(t_k, d_j) = \#(t_k, d_j) . \cdot \log \frac{|Tr|}{\#Tr(t_k)}$$

$\#(t_k, d_j)$ is the number of times the word (term) t_k occurs in the document d_j, $\#Tr(t_k)$ is the document frequency of t_k in the collection of documents Tr (i.e. the number of documents in which the term occurs) and $|Tr|$ is the size of this collection (total number of documents). The meaning of the tf-idf weighting is the following: 1) the more often a word occurs in a document, the more representative it is of the content of the document (*tf* part) and 2) if a word occurs in more documents, it is less discriminating (*idf* part), see [1], p.14.

The *tf-idf* scores are normalised between 0 and 1, e.g. using the *cosine normalisation* shown on p.12 in the Sebastiani's paper [1].

Read more about text classification, see [1], in particular the sections about bag of words, document frequency as a feature selector and tf-idf as weighting; see also [2]; it provides a clear and succinct description of various feature selection methods and is also useful as a case study in text classification.

2. Data - LingSpam-mini600 corpus

It is a subset of the original LingSpam corpus [3] and consists of 600 e-mails, each in a separate file. 200 of them are spam e-mails (the spmsg.* files) and the other 400 are legitimate e-mails. The first line of each email (after the keyword "Subject:") contains the e-mail's subject, and the remainder is the e-mail's body.

Let the **Subject corpus** be the sub-corpus containing the Subject information from all e-mails in the LingSpam-mini600 corpus. Similarly, let the **Body corpu**s be the sub-corpus containing the Body information from all e-mails in the LingSpam-mini600 corpus.

3. Data preprocessing

For each corpus (Subject and Body) use the **bag-of-words representation.**

- Preprocess the data by extracting the individual words from the corpus. Decide if you want to keep strings of numbers, punctuation symbols (e.g. ? and !) and other special symbols (e.g. $) – are they helpful in distinguishing spam from non-spam email? (Remember to justify your decision in the report).
- Remove stop-words using a stop list (there are many such lists available on the web; you can also create your own).

4. Feature selection using DF

For each corpus separately: 1) compute the DF score for each word (the result of the previous step) and 2) select the top 200 words (i.e. the ones with the highest score).

5. Feature weighting using tf-idf

For each corpus separately, apply the tf-idf weighting to the selected 200 terms, to construct the dataset that we will use to build the spam filter. Normalise the tf-idf weights using the cosine normalisation. Save the data in csv format; include a header, e.g. f1, f2, ..., f200, class. The result will be two files (one for each corpus) with 200 features and 600 examples each, where the examples are labelled as spam and non spam. Name these files as `subject.csv` and `body.csv.`

6. Implement the Naïve Bayes algorithm

As the features are numeric, you need to implement the version for numeric attributes using probability density function. Assume normal distribution, i.e. use the probability density function for normal distribution. Your program should be able to read the csv files.

7. Implement 10-fold stratified cross validation to evaluate the performance of the Naïve Bayes classifier. For each 10-fold cross validation run, your program should print the accuracy of the Naïve Bayes on the test set (the 1 fold not used for training), and at the end the average accuracy over the 10 runs.

8. Subject vs Body corpora evaluation
The goal is to evaluate the performance (accuracy) of your Naïve Bayes classifier and several other classifiers from Weka on the two corpora (Subject and Body).

- **Classification methods from WEKA** - use ZeroR, OneR, k-Nearest Neighbor (k-NN), Naive Bayes (NB), Decision Tree (DTs), Multi-Layer Perceptron (MLP) (i.e. Weka's backpropagation network) and Support Vector Machine (SVM). Use 10-fold cross validation to compare the performance of these classifiers on the two corpora.

9. Extension and challenge (worth 4 marks) – Try to improve the accuracy of your Naïve Bayes on the Body corpus
This is an open-ended part. Some ideas: 1) Stem the words after the stop-word removal, e.g. with the Porter's algorithm (there are open source implementations in different languages available on the web), 2) Instead of DF, try another feature selection method, e.g. implement Categorical Proportional Difference or one of the other methods described in [2]; 3) Conduct secondary feature selection, e.g. using the CFS method from Weka to reduce the 200 features and hopefully select a better feature subset for the Naïve Bayes, 3) Use bi-grams as features instead of 1-grams (bi-grams are 2 consecutive words). I am sure that you will have your own clever and interesting ideas!

Of course, extensions are not guaranteed to improve the result, but they will be awarded marks. In addition, the most accurate program will be awarded a prize!

10. Write a report (similar to a research paper) describing your analysis and findings. It should include the following sections

1) Aim – briefly state the aim of your study and write a paragraph why the problem is important.

2) Data preprocessing and feature selection
   - Briefly describe what you did.
   - Show the number of words for each corpus – initially and after removing the stop-words.
   - List the top 100 words for each corpus and show their DF score. Examine the list. Does the selection make sense to you given the task? Are the two lists similar?

3) Subject vs Body results and discussion

   - Results - Fill in the following tables where ZeroR, OneR, NB, DT and MLP are the Weka's classifiers tested with Weka's 10 fold cross validation; MyNB is your NB tested with your 10-fold cross validation.

Corpus: Subject	
	Accuracy [%]
ZeroR	
OneR	
1-NN	
3-NN	
NB	
DT	
MLP	
SVM	
MyNB	

Corpus: Body	
	Accuracy [%]
ZeroR	
OneR	
1-NN	
3-NN	
NB	
DT	
MLP	
SVM	
MyNB	

   - Discussion – compare the performance of the classifiers on the 2 corpora, compare your NB with the Weka's NB, anything else you consider important.

4) Challenge results and discussion– describe what you did and what your results are (accuracy using 10-fold cross validation as above) and discuss the results

5) Conclusions – summarize your main findings and, if possible, suggest future work.

6) Reflection – what was the most important thing you learned from this assignment? (1-2 paragraphs)

References:

[1] Fabrizio Sebastiani, Machine learning in automated text categorization**,** *ACM Computing Surveys*, 34(1):1-47, 2002.
[2] M. Simeon and R. Hilderman, Categorical Proportional Difference: A Feature Selection Method for Text Categorization, In *Proceedings of the Australian Data Mining Conference (AusDM)*, Adelaide, Australia, 2007.
[3] LingSpam have been collected by Ion Androutsopoulos and is described in the following paper: "An Evaluation of Naive Bayesian Anti-Spam Filtering" by I. Androutsopoulos, J. Koutsias, K.V. Chandrinos, George Paliouras, and C.D. Spyropoulos; In *Proceeding of Workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning*, Barcelona, Spain, 2000.

[1] and [2] are enclosed with the assignment specification.

COMP3608 Assignment 2 – Marking Sheet
Marked out of 20

Student(s):

	Your mark	Comments
1. [7.5 marks] Report [0.25 marks] Aim --what is the aim of the study --why is this study (the problem) important [1.5 mark] Data preprocessing– well explained --description of what was done --number of words for each corpus (initially and after stop-words removal) --top 100 words for each corpus with their DF score are shown + discussion [4 marks] Results and discussion -all results presented (for Subject and Body) -correct and deep discussion of the results ----comparison between the classifiers (accuracy, training time, other advantages) ----myNB vs Weka's NB ----comparison between corpora [1 mark] Conclusions and future work - meaningful conclusions based on the results -meaningful future work suggested [0.25 marks] Reflection (meaningful and relevant personal reflection) [0.5 marks] English and presentation --academic style, grammatical sentences, no spelling mistakes --good structure and layout; consistent formatting		
2. [3 mark] Data preparation - Splitting into 2 corpora - Bag-of-words representation (extracting words, dealing with special symbols, removing stop words) - Feature selection using DF - Feature representation using normalized tf-idf		
3. [5.5 marks] Code Stratified cross validation [2.5 marks] Naïve Bayes [3 marks]		
4. [4 marks] Extension and challenge		
Penalty: -1 mark maximum for badly written code or code that is not well documented and difficult to read		
Penalty for late submission: -1 mark for each day late		
Total (out of 20):		