I. Final Report for Interactive Learner

1. The developers

Group Number: 07

Names: Kilian Ros (s1559168) & Guido Teunissen (s1475991)

2. The Classifier: type of NBC and performance on data sets

Which type did you implement and what is the accuracy on blogs and mails?

Multinomial: **YES**

Blogs: 64.7% Without Chi2-feature selection and k-smoothing of 1. After optimization: with the Chi2 Feature selection, with a critical Chi2 value of 3.5 (found by trial-and-error), we managed to get an accuracy of 72.5%.

Mails: 98.0% without any optimization/feature selection. With the chi2-feature selection we actually got a lower accuracy, so chi2-feature selection is not recommended for this data set. Although even with a critical chi2-value of 10.84, eg. independence of 0.001 significance, we did not go below 84% accuracy, which is still much better than baseline performance.

Binomial: **NO**

Does your classifier work for any number of class values? **YES**

3. The Vocabulary: feature/word selection

What did you implement and test?

Text normalization: **YES**

Removing any special character. Only letters a-z and numbers 0-9 are not removed. Also everything is set to lowercase.

Regarding word filtering (feature selection):

Stopwords removed: **NO**

Filter words based on number of occurrences

Rare words removed: **NO**

Words that occur very often removed: **NO**

Other feature selection methods implemented: **YES: Chi2-Feature Selection**

More information about this feature selection method see appendix A.

…….

4. The Interactive Learner: the iterative strategy

A ``session'' with the IL refers to the process from starting up the IL system to finishing the program.

These questions concern the learning cycle.

- 1) Does the interactive learner (IL) **only** store the new information (documents and classes based on feedback by the user during a session) for an update at a later session? (without updating the classifier during a session) **NO**
- 2) Does the Interactive Learner update the classifier during a session? **YES**
- 3) Is the Vocabulary updated every time when a document is given a corrected class by the user during a session? **YES**
- 4) Are the probability tables updated every time when a document (or a number of documents) is given a corrected class during a session? **YES**
- 5) Can the user add new classes during a session? **YES** (*In the GUI this is possible, the TUI which was implemented before the GUI prohibits this for better UX, but it essence it is possible*)
- 6) Does the IL work for classifiers with any number of classes? **YES**

5. The User Interface and the User Instructions

GUI: **YES**

TUI: **YES**

II. Appendices

A. Chi2-Feature Selection (Bonus)

In our Interactive Learner we implemented the Chi2-feature selection method. In short the essence of this method is to pick the 'independent' words per category to keep. Independence means that the expected number of words (expected as in the number of words a category should have if the words are equally distributed), for example the word 'Flower' occurs 10x in the category 'Female', is higher or lower than the actual number of words the category has. When this is the case this word can be regarded as distinguishable for that specific category. Only the most distinguishable words per category are kept based on the critical Chi2-Value (Chi2 = 10.84 gives a significance of 0.001). (This method we use is partially inspired upon this webpage: http://nlp.stanford.edu/IR-book/html/htmledition/feature-selectionchi2-feature-selection-1.html . Accessed on December 28th 2015)

The way we implemented this method is as follows. Firstly, the normal program runs, loads in the training data set, fills the total vocabulary of all categories and the individual vocabularies. After this the Chi2-feature selection is applied. It goes through every category and every word.

Now following the example of our small data set 'book_example' we will calculate the Chi2-value manually and compare how our program calculates them. First, a general contingency table is given and then for the word 'Japan', after that the Chi2-Value will be calculated and explained with the formula.

	Word_y	**Not Word_y**	**Totals**
Category_x	**N11** = Amount of word occurrences of word_y in category_x	**N12** = Amount of word occurrences of all words that are not word_y in category_x	**Rowsum1 =** Total word occurrences in this category_x
Not Category_x	**N21** = Amount of word occurrences of word_y outside category_x	**N22** = Amount of word occurrences of all the words that are not word_y outside category_y	**Rowsum2** = Total word occurrences outside category_x
Totals	**Columnsum1 = Total word occurrences word_y**	**Columnsum2 = Total Word occurrences not word_y**	**N =** Total word occurrences in the whole classifier

With the expected number of words for each cell represented by Eij (E11, E12, E21, E22). Which is calculated by rowsum * columnsum / n. So N11 = rowsum1 * columnsum1 / N. Formula as given in the Statistics course.

Now filled in by the input of the program for the word 'Japan' in category 'H'.

	Word = Japan	Word = not Japan	Totals
Category = H	**N11** = 1 **E11** = 0.27	**N12** = 2 **E12** = 2.73	**Rowsum1** = 3
Category = Not H, so S	**N21** = 0 **E21** = 0.73	**N22** = 8 **E22** = 7.27	**Rowsum2** = 8
Totals	**Columnsum1 = 1**	**Columnsum2 = 10**	**N =** 11

When manually calculating this it is correct. The word 'Japan' only occurs in category 'H' one time, so N11 = 1. The two other words in H are Tokyo (1x) and Uganda (1x), so N12 = 2. Japan does not appear in any of the 'S' files, so N21 = 0. The total amount of words in the S files is 8, so N22 = 8 and rowsum2 = 8. The total number of word occurrences in the classifier is 11, so N=11.
The expected values Eij are also stated. For example, E11 = 1 * 3 / 11 = 0.27. (As a side note: the observed number is higher than expected, so this cell already seems to indicate independence)

Next the Chi2-value is calculated with the following formula: (from the Statistics course)

$$\chi^2 = \sum_{i=1}^{k} \frac{(N_i - EN_i)^2}{EN_i}$$

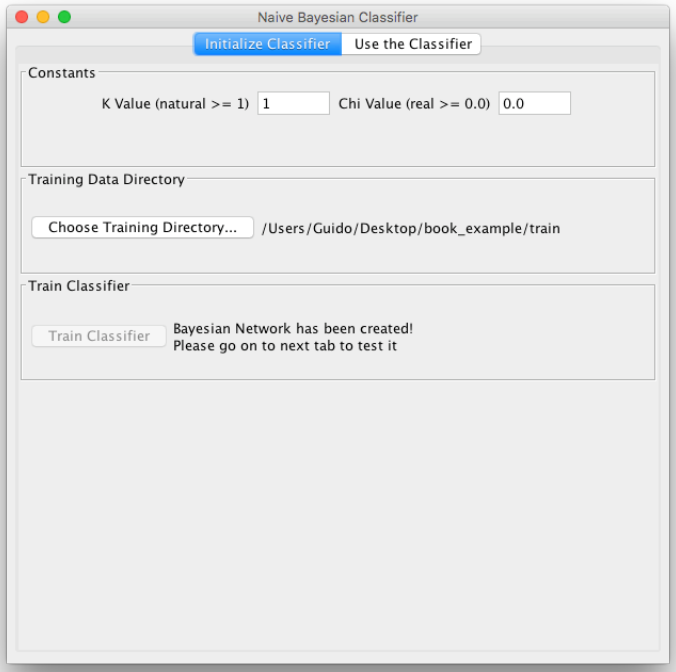
For 'Japan' in 'H' this corresponds to: Chi2 = (1-0.27)^2/0.27 + … + (8-7.27)^2/7.27 = 2.97
The program gives a value of chi2 = 2.933. This seems correct due to intermediate rounding of the numbers, which the program doesn't do.

This Chi2 value is then in our program compared to the given critical Chi2-value. The user of the program has to experiment with the Chi2-value. See accuracy rates above in section 2. Sometimes the classifier gives a much better result, and sometimes it gives a lower accuracy. It gives a lower accuracy mostly on large data sets that already had a high accuracy (>90%) without the feature selection. However, with the blogs data set we only managed to get a slightly above baseline accuracy, with the feature selection this improved to almost 75%. So we think the best way to implement Chi2 as feature selection is when the classifier gives a low accuracy already.

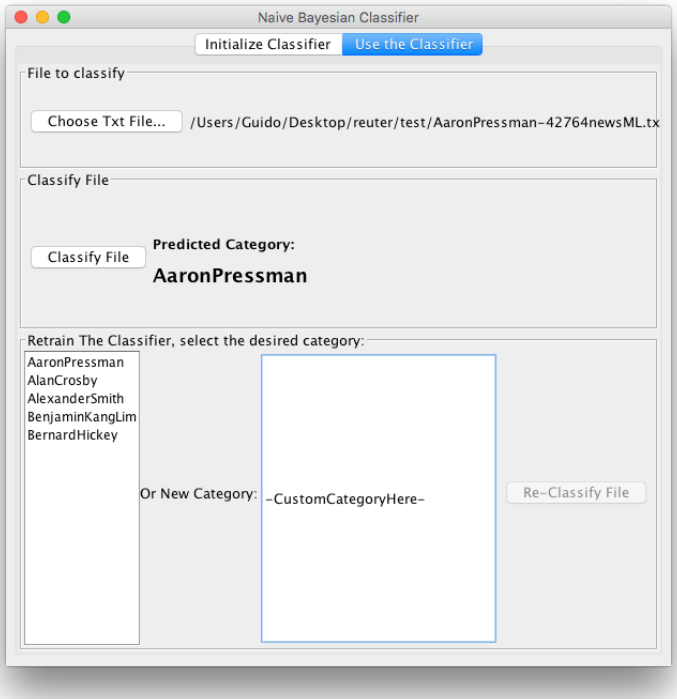
The program repeats these steps for every word and for every category and only includes the words in the vocabularies of the categories if the chi2-value is higher than the given critical value. It also updates the total vocabulary to match the newly created sub-vocabularies of the categories.

This feature selection is applied directly after the initial training; after loading in all the normalized words from the txt files and creating vocabularies/bag-of-words and their corresponding categories. It is also applied every time when the system is retrained by the user.

B. GUI



First and second tab of the GUI.



C. Additional Notes about the program

General overview of the program

The general overview of the program is the following. Basically the whole network is an instance of the class 'Categories' which represents a collection of categories (eg. SPAM or HAM, etc.), and a total vocabulary (word name and word count). The Categories class has some helper methods to each time calculate all the probabilities and picks out the maximum value. The Categories class has many Category instances, also with each their own Vocabulary. Other classes are helpers.

The program starts by going through each of the files in the given directory, normalizing every word and converting this to a Java data type. For each file the corresponding words and their word count are added to the corresponding category's vocabulary (if the category does not exist yet, a new one is made based on the name of the file).
After every file has been imported, chi2 feature selection is applied. This is mentioned in Appendix A.

The program Retraining works in almost the same way as the start of the program. It basically adds the words to the category's (new or existing, based on the name the user gives) vocabulary and when calculating the new word counts are used. Also feature selection is applied after every re-training.

The File Reader identifies the category of a file only by the first word that is separated by a minus sign. Format: CATEGORY-restFileName.txt. Eg. SPAM-training123.txt. Please keep this in mind. They also should be in one and the same folder. It does not matter where the folder is located.

Retraining example

For this example, showing that the classifier learns from input, the book_example data set is used. Separated in a train set with two categories 'H' and 'S', and a test set where we will introduce a new category of 'STAR' (which represents something like mailing a friend, which has some similar words in them). The folder 'train' is used for training the classifier.

The STAR-1.txt file will be provided to the system as input. All the three STAR-files are classified as 'H' right after the initial training. The following happens in out program:

STAR-1.txt -> predicted category by classifier: 'H'. User input: new custom category of 'STAR'
STAR-2.txt -> predicted category by classifier: 'STAR'.
STAR-3.txt -> Also a prediction of 'STAR'

This not only shows the classifier can predict a category based on user input, which it did not do before, but is also able to accept input and then recognize a newly created category (STAR).

Constants declaration

We urge the user to experiment with the critical chi2-value with a train and test set and measure the accuracies (in TestPredictionAccuracy.java) without chi2 (critical value = 0.0), 0.001 independence significance (critical value of 10.84) and in between these values. For example, we observed a critical value of 3.5 to be the best for the blogs data set.

The k-smoothing value has a much lower effect when changing its value and can even be neglected and set to 1.

Extra Data Set 'Reuter'

To do another check than the standard given data sets on Blackboard, also a part of the *Reuter* data set was used. (Downloaded from: https://archive.ics.uci.edu/ml/datasets/Reuter_50_50 on January 3rd 2016).
The performance (with k=1 and Chi2 = 0.0) was very high. An accuracy of 95.2%. See the folder 'reuter' for the used data. A higher chi2 value was not desirable, therefore it was kept at 0 for best performance.