Classification of News Articles

Machine Learning Project

Jakob Sahlström
jasa5691@student.uu.se

Anders Lindström
anli6945@student.uu.se

Jesper Dürebrandt
jedu6357@student.uu.se

Uppsala University

Sunday 1st June, 2014

Abstract

Four classifiers, namely Bernoulli Naïve Bayes, Multinomial Naïve Bayes, Random Forest, and Support Vector Machine, were investigated in the field of text classification. The task was to classify the topics of news articles. The main focus of this report was to analyze, and compare between classifiers, the performance impact of varying vocabulary size due to feature selection for different data types as well as difficulties of distinguishing the eight different topics. All classifiers were anticipated to perform well in this task, but Multinomial Naïve Bayes proved to be the best classifier with a maximum hit ratio of 76% with a vocabulary size of 1022 words.

1 Introduction

There are many different types of classifiers and they all have their advantages and disadvantages. This project investigates the performance of four classifiers that are common in the field of text classification, namely Bernoulli Naïve Bayes, Multinomial Naïve Bayes, Random Forest, and Support Vector Machine. The investigation is carried out by giving the classifiers the task of classifying news articles from eight different topics and then comparing the results.

2 Related Work

2.1 Naïve Bayes Classifier

Naïve Bayes models are widely used because of their simplicity and efficiency. A. McCallum and K. Nigam compares the two most common models, the multi-variate Bernoulli and the Multinomial model, in the realm of document classification. They are explained in detail both theoretically and empirically and in general the Multinomial model outperforms the Bernoulli model. [8]

2.2 Random Forest Classifier

Random Forest is a decision tree based classification model. It has been used to classify web documents by keywords, where it for five and seven topics performed better than, e.g., Naïve Bayes and Multilayer Perceptron [4]. In the paper by Koprinska et al. they use Random Forest for classifying e-mails. Random Forest was able to out perform other methods, such as Decision Tree, Support Vector Machine and Naïve Bayes. [5]

2.3 Support Vector Machine Classifier

Support Vector Machine (SVM) classifiers perform well on data that is linearly separable, since they find the optimal hyperplane that separates the data. They can however only separate data into two classes, but if combined they are able to perform multi-class classification. A simple approach is to use k SVMs to solve a k-class classification problem. The SVMs may also be combined in a more sophisticated fashion, so that less than k SVMs can be used. [7]

2.4 Data Preparation & Representation

Tampakas et al. discusses different kinds of data preparation and preprocessing, such as stemming and stop word deletion, which can be used to reduce the feature set and improve performance of the classifier. They also discuss various ways of representing the data. We have decided to follow the outline of the text classification process they present and investigate the effect of different kinds of data representation. [6]

3 Theory

This section introduces the theory used throughout the project. For more detailed information, see the provided references.

3.1 Naïve Bayes Classifier

Let c be the class and $A = a_1, \ldots, a_n$ be the attributes of a document. Then Bayes Theorem states that

$$p(c|A) = \frac{p(A|c)p(c)}{p(A)}. \quad (1)$$

The attributes A are classified as class c if and only if

$$f_b(A) = \frac{p(c|A)}{p(\neg c|A)} \geq 1, \quad (2)$$

where $f_b(A)$ is called a *Bayesian* classifier. If assuming all attributes are independent given the class, $p(A|c)$ can be rewritten as

$$p(A|c) = p(a_1, \ldots, a_n|c) = \prod_{i=1}^{n} p(a_i|c).$$

Now the final classifier can be written as

$$f_{nb}(A) = \frac{p(c)}{p(\neg c)} \prod_{i=1}^{n} \frac{p(a_i|c)}{p(a_i|\neg c)} \quad (3)$$

where f_{nb} is called the *Naïve Bayesian* (NB) classifier and is a *binary classifier*, separating two classes.

The assumption that, given a class the attributes are independent, means that the distributions of each attribute can be estimated as a one dimensional distribution. This reduces the effect of the curse of dimensionality. In general a Naïve Bayes classifier is a good predictor but a bad estimator. [11]

Two models that use the Naïve Bayes assumption are the *multi-variate Bernoulli* model and the *Multinomial* model. The main difference is that they use different assumptions regarding the distribution of $p(a_i|c)$. In the Bernoulli model the attributes are binary, indicating if a word from a vocabulary has occurred at least once or not, i.e. $p(a_i|c)$ has a Bernoulli distribution. In the multinomial model the frequency of words are taken into account. This leads to a multinomial distribution of $p(a_i|c)$. [8][9][10]

3.2 Random Forest Classifier

Random Forest (RF) is based on building several small decision trees, as shown in Algorithm 1.

Algorithm 1 Random Forest

```
Let $x = (x_1, x_2, ..., x_N)$ be a set of features;
while not enough trees do
    Randomly pick with replacement a subset of $n \ll N$ features;
    Use training set to build a decision tree using a classification algorithm, e.g. C4.5, without pruning.
end while
```

After all the trees are built, they can be used to let a new vector of data pass through all the trees and then letting each tree *vote* on what class the vector most probably should be a part of. [1]

3.3 Support Vector Machine Classifier

Support Vector Machines (SVMs) classify data belonging to two classes $y_i = \pm 1$ by finding the hyperplane $\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$ with the widest margin that separates the classes in the training data $\boldsymbol{x}_i$. The constant b is the hyperplane's offset from the origin, $\boldsymbol{x}$ are vectors located within the hyperplane and the normal to the hyperplane, and the weights $\boldsymbol{w}$ determine the orientation. The scale of $\boldsymbol{w}$ and b is defined such that $\boldsymbol{w} \cdot \boldsymbol{x} + b = +1$ and $\boldsymbol{w} \cdot \boldsymbol{x} + b = -1$ for the vectors that restrict the margin of the hyperplane on the left and right side respectively. Those vectors are referred to as support vectors and they fully specify the decision function

$$f(\boldsymbol{x}) = \text{sign}(\boldsymbol{w} \cdot \boldsymbol{x} + b).$$

Finding the optimal hyperplane is a maximization problem, where the objective function describes the width of the margin $\frac{1}{||\boldsymbol{w}||_2}$, which is equivalent to minimizing $\frac{1}{2}||\boldsymbol{w}||_2^2$ subject to the constraints $y_i(\boldsymbol{w} \cdot \boldsymbol{x} + b) \leq 1, \forall i$. The problem can be formulated using Lagrangian multipliers $\alpha_i \geq 0$ into

$$L(\boldsymbol{w}, b) = \frac{1}{2}(\boldsymbol{w} \cdot \boldsymbol{w}) - \sum_i \alpha_i(y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) - 1).$$

Using that the derivatives with respect to $\boldsymbol{w}$ and b are zero at the minimum, we can express $\boldsymbol{w}$ in terms of $\boldsymbol{x}_i, \alpha_i$, and y_i and then reformulate the problem into maximizing

$$W(\alpha) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j (\boldsymbol{x}_i \cdot \boldsymbol{x}_j), \quad (4)$$

with respect to α_i and subject to the constraints

$$\alpha_i \geq 0, \quad \sum_i \alpha_i y_i = 0,$$

which is solved using quadratic programming. An advantage with Support Vector Machines is that the maximization problem is convex, meaning that the maximum found is guaranteed to be the global maximum. This requires, however, that the classes are linearly separable. If they are not, one can make use of a so called *kernel function* $\boldsymbol{\phi}(\boldsymbol{x})$ that maps the data onto a space in which the classes are linearly separable. This is done by substituting the dot product $\boldsymbol{x}_i \cdot \boldsymbol{x}_j$ in equation (4) with $\boldsymbol{\phi}(\boldsymbol{x}_i) \cdot \boldsymbol{\phi}(\boldsymbol{x}_j)$. Support Vector Machine are suitable for problems with high dimensionality, but might perform poorly if the number of samples is much smaller than the number of dimensions. [2]

3.4 Multi-Class Problem

Classifiers like Naïve Bayes and Support Vector Machine are binary classifiers. For a multi-class problem the strategy *One-Vs-The-Rest* is used which creates one classifier for each target class. Then for each classifier fit the feature vector to that class against not that class. The classifier with highest confidence will be the final class of the feature vector.

3.5 Feature Selection

Features that provide more information than others can be emphasized by feature selection. One way of assigning a score to each feature is to use the χ^2 function, which is a measure of how independent a feature is of the classes. The lower the score, the more independent is the feature of the classes. Features with high scores are therefore desired. The χ_j^2 score for feature j is given by

$$\chi_j^2 = \sum_{i=1}^{N_c} \frac{(O_{i,j} - E_{i,j})^2}{E_{i,j}},$$

where N_c is the number of classes, $O_{i,j}$ is the count of feature j in class i, and $E_{i,j}$ is the expected count given by

$$E_{i,j} = \frac{N_{s,i}}{N_s} \sum_{n_c=1}^{N_c} O_{n_c,j},$$

where N_s is the number of samples, $N_{s,i}$ is the number of samples of class i and the sum is the total count of feature j.

4 Method

This section describes the method of investigating the different classifiers, how the data is processed and the settings used when classifying. Figure 1 illustrates the outline of the process.

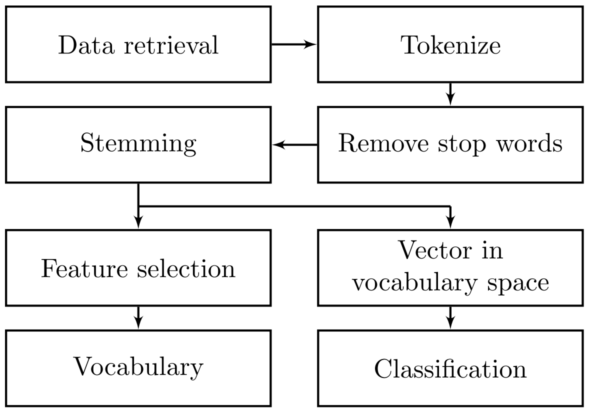


Figure 1: The process of making the vocabulary and a feature vector. The vocabulary reads the whole training set and the feature vector reads one article in the test set.

4.1 Data Retrieval

A crawler was implemented to extract articles from BBC's RSS feed[1]. The crawler extracts the content of each article from eight different topics, namely

- Business
- Education
- Entertainment and Art
- Health
- Politics
- Science and Environment
- Sports
- Technology.

A total of 734 articles were extracted. As training set 2/3 of the articles were randomly selected and the remaining 1/3 were used as test set. Notable is that the distribution of topics is not uniform since the RSS feeds of the topics are not updated equally frequent.

4.2 Coding

Python is chosen as programming language in order to extend our knowledge and experience of implementing different types of classifiers using the Scikit-learn machine learning library in Python and further understand how the data has to be prepared and presented to the classifiers.

4.3 Preparation of Data

To classify the articles as accurately and fast as possible, the data is prepared in such way that it contains as much information as possible in a format as dense as possible. To achieve this, the articles are parsed into words with white-space as separator, then removed of all characters not found in the English alphabet, though keeping the '-' character. All so called stop words are removed (such as "a", "the", etc.), and the words are stemmed so that similar words would be on the same form, and not be seen as two different words. (E.g., "argued", "argues", "arguing" are reduced to "argu".)

4.4 Vocabulary

The vocabulary is built up by unique stemmed words from the training set. By different pruning techniques the vocabulary is reduced in size and only the most informative elements are kept. For the feature selection the score is determined by the χ^2 function. The vocabulary is then pruned by selecting a percentile of the best features.

4.5 Data Types

There are many different ways to represent a feature vector, and different classifiers might want it represented in different ways. We have chosen to investigate a couple of different types of representations of the vocabulary. The data types are

Binary: Does article contain the word or not.
Count: How many times does article contain the word.
L_2-normalized: The count array normalized with the L_2 norm.
Mapped value from 0 to 1: $\frac{c_i}{\max(c)}$ where c_i is the count of word i in an article.

4.6 Classifiers

In the Scikit-learn package[2] there are plenty of different classifiers, amongst them the four we want to investigate. The four classifiers we will be investigating are

- Naïve Bayes: Bernoulli
- Naïve Bayes: Multinomial
- Random Forest
- Support Vector Machines

4.6.1 Bernoulli Naïve Bayes

The settings for the Bernoulli Naïve Bayes classifier in the Scikit-learn package throughout the tests, if nothing else is stated, has been

Alpha:	1.0
Binarize:	0.0
Fit prior:	True
Class prior:	None

For more information about the different settings, see description of the Bernoulli Naïve Bayes classifier in the Scikit-learn package.

[1]`http://www.bbc.com/news/10628494`

[2]`http://scikit-learn.org/`

4.6.2 Multinomial Naïve Bayes

The settings for the Multinomial Naïve Bayes classifier in the Scikit-learn package throughout the tests, if nothing else is stated, has been

Alpha: 1.0
Fit prior: True
Class prior: None

For more information about the different settings, see description of the Multinomial Naïve Bayes classifier in the Scikit-learn package.

4.6.3 Random Forest

The settings for the Random Forest classifier in the Scikit-learn package throughout the tests, if nothing else is stated, has been

Number of trees: 500
Criteria: Gini impurity and entropy
Max features per tree: $\sqrt{\text{number of features}}$
Max depth: None
Minimum of samples required to split: 2
Bootstrap: True

For more information about the different settings, see description of the Random Forest classifier in the Scikit-learn package.

4.6.4 Support Vector Machine

The settings for the Support Vector Machine classifier in the Scikit-learn package throughout the tests, if nothing else is stated, has been

Penalty parameter C: 1.0
Kernel: Linear
Probability estimates: False
Shrinking heuristic: True
Tolerance for stopping: 1e-3
Class weight: Auto (adjust weights inversely proportional to class frequencies)
Hard limit on iterations within solver: No limit
Random state for seed to random generator: None

For more information about the different settings, see description of the Support Vector Machine classifier in the Scikit-learn package.

4.6.5 Hybrid

For the Hybrid classifier, we let each of the four classifiers classify the sample, and then let them *vote* for which class the sample should belong to. The Hybrid classifier chooses the majority to predict the class. We chose to have the prediction of the Multinomial Naïve Bayes as default in the case of a tie.

4.7 Investigation

In order to compare the different classifiers and test their performance, the hit ratios will be investigated by varying the vocabulary size and the amount of articles in the training data for the different data types. Confusion matrices and results of how similarly the classifiers classify the articles might reveal strength and weaknesses of the classifiers as well as difficulties with the data.

5 Results

The result of the analysis can be seen in Figure 2 - 4 and Table 1. Figure 2 shows, for the different models and data types, the overall hit ratio of correct classification as a function of vocabulary size after χ^2 pruning. By comparing the figures it can be noted that the maximum accuracy of 76% is attained with binary inputs by Multinomial at a vocabulary size of 1022 words. Another observation is that Bernoulli is sensitive to vocabulary size compared to other models and Support Vector Machine seems in contrast to Bernoulli more robust to changes in vocabulary size. More results derived from these figures are that Random Forest is independent of the feature format and Hybrid follows the behavior of Multinomial.

The confusion matrices in Figure 3 describes the accuracy for specific topics. The rightmost column describes for a certain topic how many articles it contains and the accuracy of the topic. The bottom row describes for a certain predicted class, how many times that was the target class, and the accuracy when predicted. One can observe that Sports and Education are fairly easy to classify in contrast to Technology which is often misclassified as Science & Environment.

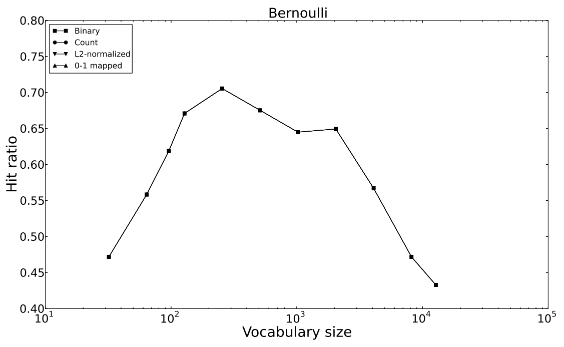
Notable is that Random Forest has a tendency of predicting Sports.

Figure 4 shows, for each model, how the hit rate depends on number of articles used as training set. For each selected subset of size 6 to 503 articles the 500 features with the highest χ^2 scores are kept in the vocabulary and used to train the classifiers in order to keep the dimension of the feature space constant.

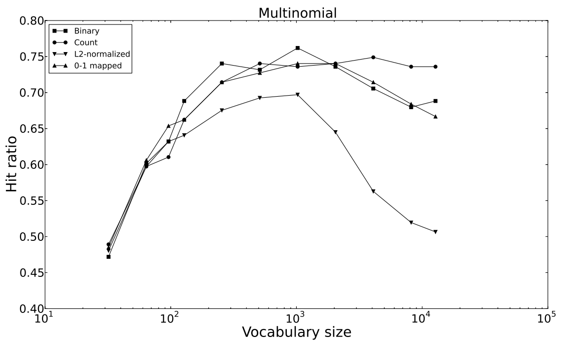
In Table 1 a comparison of the similarity of prediction when mispredicting is shown. The values resembles, given that both classifiers predicted wrong class, how many times they predicted the same class.

Table 1: Percentage that two classifiers, when both classifying wrong, classifies test to the same class. A total of 231 articles were tested. A vocabulary size of 511 words and the data type *"Mapped value from 0 to 1"* was used.

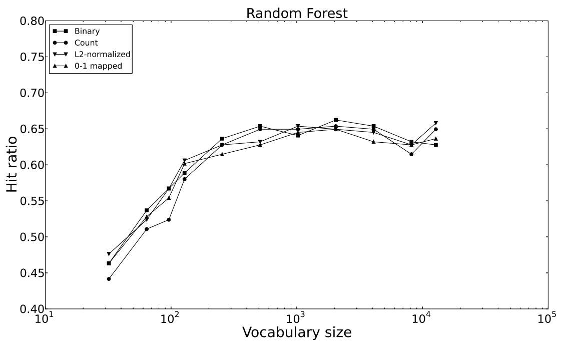
	Bernoulli	Multin.	RF	SVM
Bernoulli	100%	67.3%	69.0%	58.8%
Multin.	67.3%	100%	78.6%	76.9%
RF	69.0%	78.6%	100%	75.4%
SVM	58.8%	76.9%	75.4%	100%



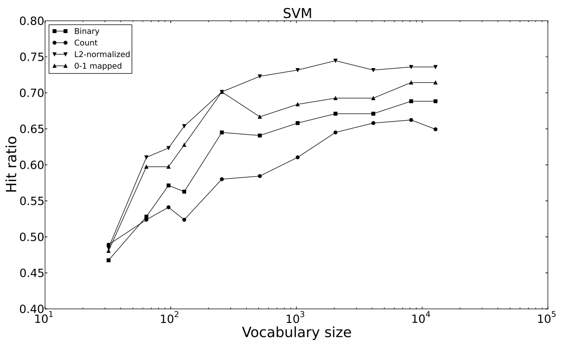
(a) Hit ratio of Bernoulli classifier with varying vocabulary size. All values greater than zero are mapped to one, hence the different data types result in the same accuracy.



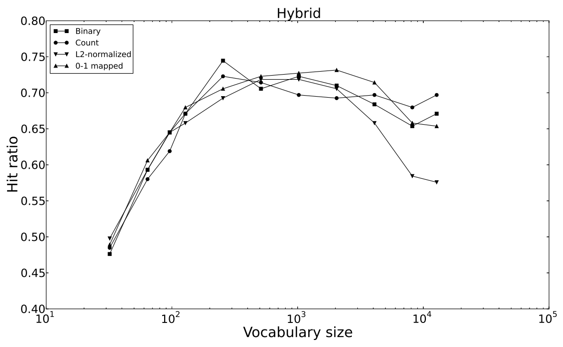
(b) Hit ratio of Multinomial classifier with varying vocabulary size.



(c) Hit ratio of Random Forest classifier with varying vocabulary size.

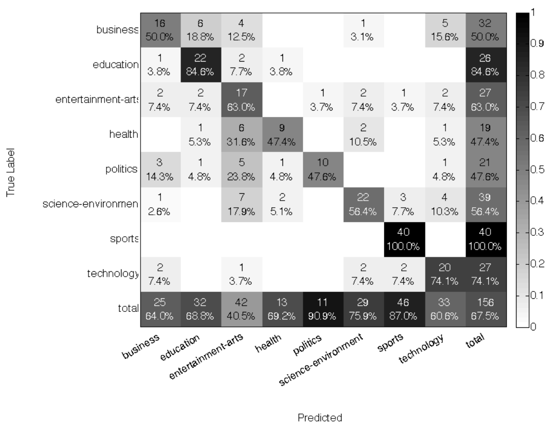


(d) Hit ratio of SVM classifier with varying vocabulary size.

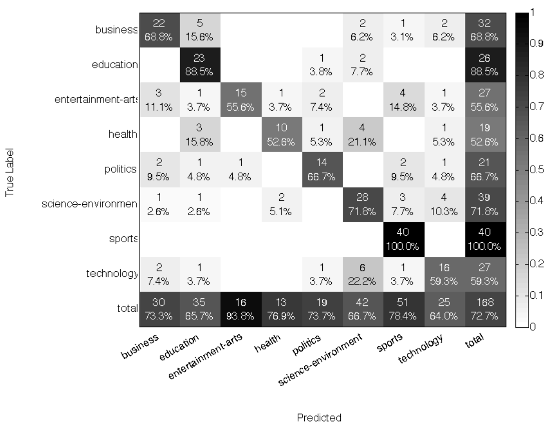


(e) Hit ratio of Hybrid classifier with varying vocabulary size.

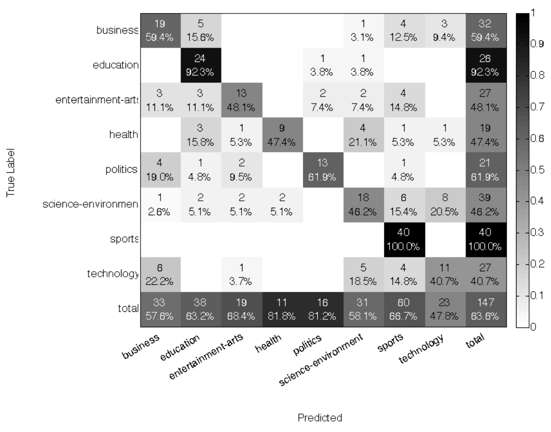
Figure 2: Hit ratio as a function of vocabulary size after χ^2 feature selection for different classifiers and data types.



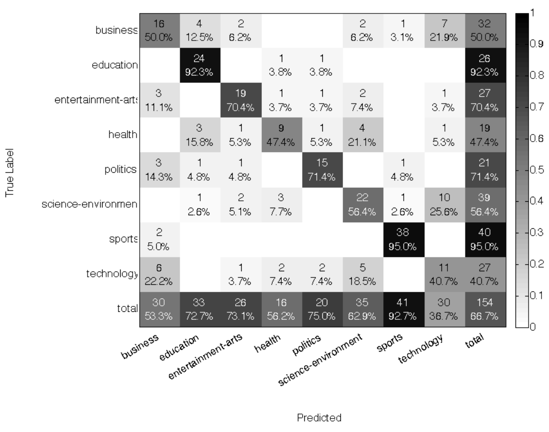
(a) Confusion matrix of Bernoulli.



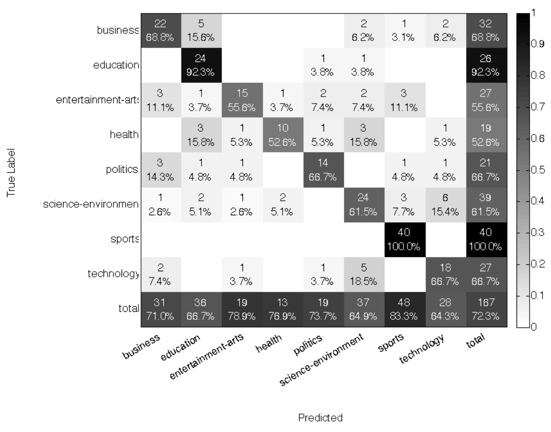
(b) Confusion matrix of Multinomial.



(c) Confusion matrix of Random Forest.

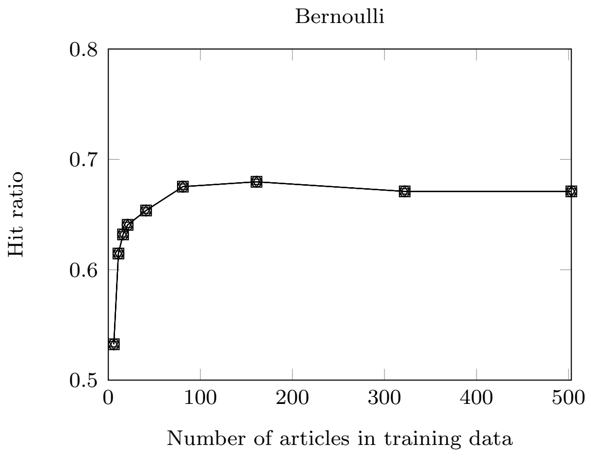


(d) Confusion matrix of SVM.

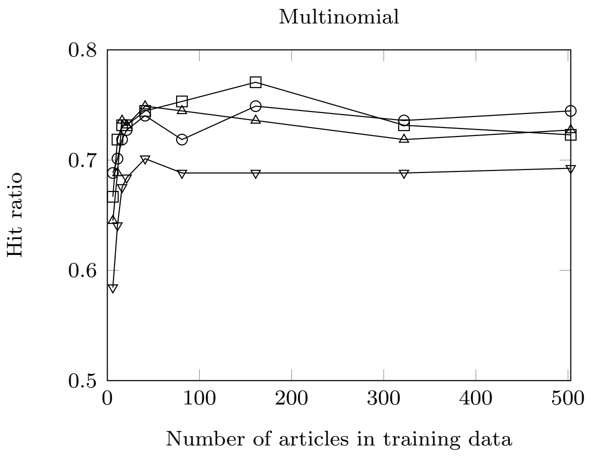


(e) Confusion matrix of Hybrid.

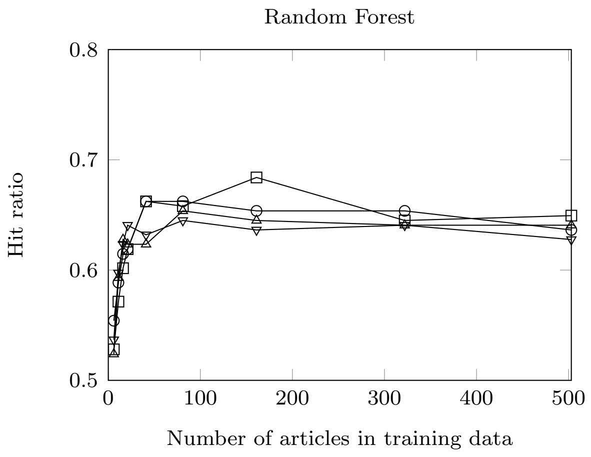
Figure 3: Confusion matrices for the different classifiers. A total of 231 articles were tested. A vocabulary size of 511 words and the data type *"Mapped value from 0 to 1"* was used. The rightmost column describes for a certain topic how many articles it contains and the percentage of how correct the model predicted. The bottom row describes for a certain predicted class, how many times that was the target class, and the percentage correct out of them.



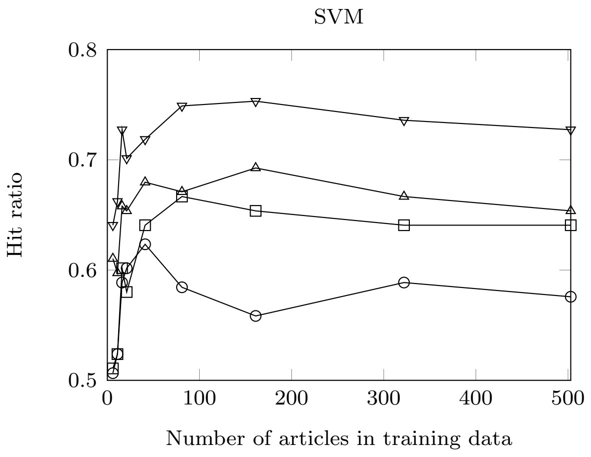
(a) Hit ratio of Bernoulli classifier with varying number of articles in training data. All values greater than zero are mapped to one, hence the different data types result in the same accuracy.



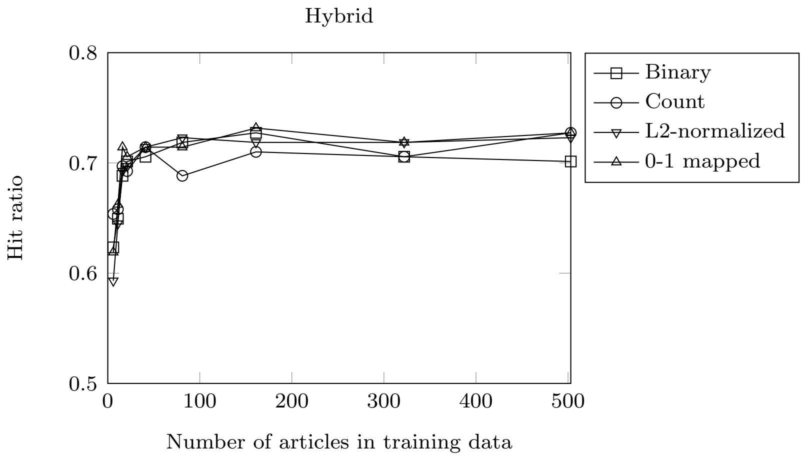
(b) Hit ratio of Multinomial classifier with varying number of articles in training data.



(c) Hit ratio of Random Forest classifier with varying number of articles in training data.



(d) Hit ratio of SVM classifier with varying number of articles in training data.



(e) Hit ratio of Hybrid classifier with varying number of articles in training data.

Figure 4: Hit ratio as a function of number of articles in training data for different classifiers and data types at a constant vocabulary size of 500 features after χ^2 selection.

6 Discussion

Interesting results and aspects of the report will be discussed further under this section, where we will give our analysis and input on *why* we think the results turned out how they did. Later, under Section 7, conclusions will be drawn, along with suggestions on what future projects should focus upon.

6.1 Naïve Bayes

Comparing the two models using Naïve Bayes assumption in Figure 2a & 2b our results clearly shows that the Bernoulli model can not handle a large vocabulary size which the Multinomial model clearly can. Even though they both make the naïve assumption of independent features they do have different assumptions regarding the distribution within the features. Why Multinomial works so well is probably because it's assumption is close to the reality.

6.2 Support Vector Machine

In Figure 2d we see that Support Vector Machine is resistant to increasing number of features. This follows the properties of the classifier mentioned in Section 3.3 saying that Support Vector Machines are suitable for problems with high dimensionality. Different Kernel functions, namely radial basis function and polynomial, were tested but not investigated further since the linear Kernel function gave satisfactory results.

6.3 Random Forest

Why the Random Forest classifier is similar for all four types of data in Figure 2c is because it splits the data into two parts in each node depending on threshold values, so if the data is normalized or not, when building the tree, does not really matter for the classifier. We expected to see more difference in the binary data type, due to the fact that it discards how *many* of each word the article contains.

One of the reason to why the Random Forest classifier tends to classify too many as Sports is that Random Forest is sensitive to imbalanced training data, i.e., not having the same amount of training data for each class. In our case, we have more articles of the class Sports, and less of Politics and Health. The classification in Figure 3 follows this reasoning and shows that Sports was classified 50 % more than there are Sports articles, while Politics and Health were classified 23 % respectively 42 % less than there are articles of those classes.

6.4 Hybrid

In Figure 2 & 3 we can see that the Hybrid classifier follows the trend of Multinomial, which is due to the fact that Multinomial is set as default in case of a tie in the voting.

6.5 Data and Feature Selection

The feature selection used is based on a statistical approach; the features that are statistically dependent of the classes are rewarded a high score and are therefore kept. This is reasonable since the features that are independent of the classes does not contribute with information to the classification. It does not, however, take feature interaction into account. It might be the case that a group of features with lower scores might give more information than a feature with a high score.

As illustrated in Figure 4, the hit ratio as a function of articles in the training set has a similar behavior for all the classifiers. In general, there is an increase of the hit ratio between zero and 100 articles, and for more than 100 articles the hit ratio is constant or decreases slightly. The hit ratio peak is reached for a relatively small number of articles; considering that there are eight different article classes, there are only a few articles per class. This could be explained by the fact that all the articles are from the same point in time, and therefore cover very similar or the same events. If all articles contain very distinguishing features, these features will be found even in a small subset of the training data.

6.6 Classification Difficulties

As stated in Section 5 the topics Technology and Science & Environment has a tendency of getting mixed up by all classifiers. This is not a big surprise since in nature these two topics are closely related, e.g., an article about a new technology that reduces the environmental footprint at a technology cooperation could legitimately be classified as either of the two topics.

Why Sports is easy to classify may depend on the fact that we attained more articles about Sports than any other topic which made the classifiers have a prior probability to predict Sports and also more data to classify it with. Sports also uses a quite unique vocabulary compared to other topics since Sports articles' target audience is those who are already interested in the topic in contrast to other topics which aims to reach an as big audience as possible.

Even a person might have difficulties to classify an article after reading it. A reason for this might be that the particular article does not contain enough information in order to distinguish the topic, which makes the classification difficult for a person as well. Imagine for example a football player, visiting a school to talk about politics.

Since the content of news articles changes with time, one might need data from several points in time in order to get a good generalization. For a human, a politics article might be distinguishable from a sports article due to the fact that we probably know more or less all words, but for our classifier, this will be a harder task if it has never been exposed to those words.

7 Conclusions & Future Work

Multinomial Naïve Bayes proved to be the most suitable classifier for the task of classifying news articles of eight different classes, since it obtained the highest hit ratio and had a good overall performance for different datatypes.

Support Vector Machine proved to be the classifier that is most sensitive of the data type. Figures 2 and 4 show that the performance is poor for the data type count and that the performance is best for the L_2-normalized data. There are however many parameters, such as Kernel function, penalty parameter C, and tolerance for stopping, for this classifier that can be fine-tuned and maybe yield a better result.

It would be of interest as future work to do an investigation of how the Random Forest classifier behaves if the data is evenly balanced between the classes. For further reading in the field of very imbalanced data, see [3].

Adding document characteristics as features, such as average word length and number of digits, and analyzing other pruning techniques, e.g. TF-IDF which emphasizes words that are unique for a document and suppress the value of common words among all documents, would also be an interesting extension to this report.

It would be interesting to see how the hit ratio would turn out if a human was to classify 100 articles from these eight topics.

References

[1] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[2] C. Campbell and Y. Ying. *Learning with Support Vector Machines.* Morgan & Claypool Publishers, 2011. `http://www.morganclaypool.com/doi/pdf/10.2200/S00324ED1V01Y201102AIM010`. Last checked 24 May, 2014.

[3] Chen Chao, Andy Liaw, and Leo Breiman. Using random forest to learn imbalanced data. Technical report, Statistics Department, University of California at Berkeley, 2004. `http://digitalassets.lib.berkeley.edu/sdtr/ucb/text/666.pdf`. Last checked 29 May, 2014.

[4] Myungsook Klassen and Nikhila Paturi. Web document classification by keywords using random forests. In Filip Zavoral, Jakub Yaghob, Pit Pichappan, and Eyas El-Qawasmeh, editors, *Networked Digital Technologies*, volume 88 of *Communications in*

Computer and Information Science, pages 256–261. Springer Berlin Heidelberg, 2010.

[5] Irena Koprinska, Josiah Poon, James Clark, and Jason Chan. Learning to classify e-mail. *Inf. Sci.*, 177(10):2167–2187, May 2007.

[6] V. Tampakas M. Ikonomakis, S. Kotsiantis. Text classification using machine learning techniques. *Wseas transactions on computers*, 4(8):966–974, 2005.

[7] Eddy Mayoraz and Ethem Alpaydin. Support vector machines for multi-class classification. In José Mira and JuanV. Sánchez-Andrés, editors, *Engineering Applications of Bio-Inspired Artificial Neural Networks*, volume 1607 of *Lecture Notes in Computer Science*, pages 833–842. Springer Berlin Heidelberg, 1999.

[8] Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. AAAI-98 workshop on learning for text categorization, 1998. `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.65.9324&rep=rep1&type=pdf`. Last checked 29 May, 2014.

[9] Eric W. Weisstein. Bernoulli distribution. From *MathWorld* - A Wolfram Web Resourse., 2008. `http://mathworld.wolfram.com/BernoulliDistribution.html`. Last checked 29 May, 2014.

[10] Eric W. Weisstein. Multinomial Distribution. From *MathWorld* - A Wolfram Web Resourse., 2008. `http://mathworld.wolfram.com/MultinomialDistribution.html`. Last checked 29 May, 2014.

[11] Harry Zhang. The optimality of naive bayes. FLAIRS2004 conference, 2004. `http://www.aaai.org/Papers/FLAIRS/2004/Flairs04-097.pdf`. Last checked 29 May, 2014.