# Text categorization on Reuters corpus

**Ivana Lukšová**

Introduction to Machine Learning, 2014

## 1 Task

The task of text categorization can be described as follows: given a set of documents, we want to assign to each document one or more text categories or no category.

In this term project, we want categorize documents from the well-known Reuters-21578 corpus which is a collection of 21578 articles published on Reuters in 1987.

We have chosen only three most frequent text categories as the target categories:

Mergers/Acquisitions (ACQ)
Earnings and Earnings Forecasts (EARN)
Money/Foreign Exchange (MONEY-FX)

Since this categories may overlap, we have at total **8 target labels** for each subset of these categories. We want to train a classifier that will assign a target label to a given document.

## 2 Approach

In the text categorization task, we want to compare two different approaches. Both approaches are based on building an ensemble of binary classifiers. In the first approach, we will use a simple **majority voting** method to assign a target label. In the second approach, we will use the outputs of this ensample as the input for **another classifier**.

Our ensemble of classifiers will consist of 8 classifiers: for each text category (ACQ, EARN, MONEY-FX) we will train 3 binary classifiers with different machine learning methods:

- Support Vector Machine (SVM)
- Random Forest (RF)
- Naïve Bayes (NB)

The output of each of these binary classifiers will be TRUE or FALSE – assigning or not assigning a given label.

In the majority voting approach, if two of classifiers will output TRUE, we will assign the particular text category to an input document. The target label will be determined as the composition of these assigned text categories.

In the second approach, we will train a decision tree that will use these eight binary outputs. The decision tree will directly assign the target label.

### 2.1 Features

In the text categorization task, the first step is to transform text documents into a set of features suitable for classifier learning methods. Convenient transformation could be representing the document as a set of words, ignoring their order in the text. Thus we will split the text documents into the words and then each feature will correspond to occurrence of a particular term in the text.

Thus the feature values can be represented as a **term-document matrix** – a matrix which rows describes the documents and the columns corresponds to particular terms.

There are several ways how to determine the values of entries in the term-document matrix. We can use following metric:

| Metric name | Value | Description |
|---|---|---|
| **Term occurrence** | TRUE/FALSE | Describes the occurrence/unoccurrence of a term in the document |
| **Term frequency** | Integer number | Describes how many time a particular term appears in the document |
| **Term frequency – inverse document frequency** | Real number | The term frequency is offset by its frequency in the corpus |

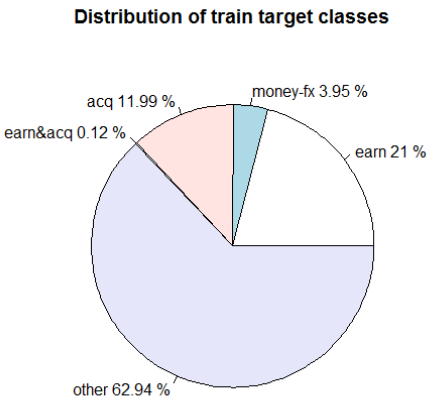
This approach could lead to huge feature space. To decrease computational complexity and avoid related issues such as overfitting, we want to use statistical methods to select only important words and filter unimportant words. It is evident that the most frequent words (STOP words), such as *"the"*, *"and"*, "he", do not bring any information about the text category. We want to remove also very infrequent words in the texts – if a word appears only few time in the corpus, it is not the useful one.

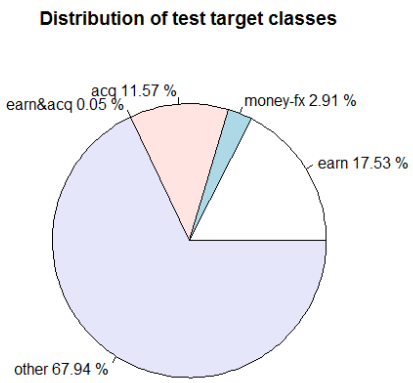
Another improvement can be done by term stemming - "merging" multiple forms of one word to one. For example, an occurrence word "*accounts"* can be merged into an occurrence of term *"account"*. We suppose that this approach can improve the performance of the output classifier.

# 3 Data

As we mentioned earlier, the Reuters-21578 corpus consists of 21578 articles. This corpus is distributed in a SQML or XML form and the documents are already divided into several groups – splits. In this term project, we will use the **Lewis split** that divides this documents into the training and testing set. The training set contains 13624 documents and the testing set contains 6188 documents. The distribution of text categories in these sets is as follows:

| | Training set | Test set |
|---|---|---|
| **EARN** | 2861 | 1085 |
| **ACQ** | 1633 | 716 |
| **MONEY-FX** | 538 | 180 |
| **EARN & ACQ** | 16 | 3 |
| **Other** | 8576 | 4204 |
| Total | **13624** | **6188** |




We see that we have only 5 target labels, because there is no combination of multiple labels except the EARN & ACQ.

# 4 Implementation

In this section, we will describe the details of the implementation of our approach in the text categorization task.

## 4.1 Data preparation

XML format of the corpus is not convenient for textual and statistical processing. Thus the first step is to process the corpus and to extract text documents along with their categories and store it in more convenient format.

This involves also first preprocessing of the text – we remove the numbers and other non-alphabetic characters and the text is transformed to lower case letters. Texts along with their categories is stored in the `data.frame` representation.

## 4.2 Creation of term-document matrix

We have used the `tm` library to create and represent the term document matrix. With this library, we are able to perform statistical filtering such as removing the STOP words, words steming and to select the metric to represent the values in the matrix.

Example of usage of the `tm` library:

```
control=list(bounds=list(local=c(1,Inf)),language="english",stopwords=TRUE,
wordLengths=c(3,Inf), weighting=weightTf)

tdm=DocumentTermMatrix(corpus,control=control)
```

This library allows us to remove also the very unfrequent words by the controling the `sparse` parameter:

```
tdm=removeSparseTerms(tdm, 0.40)
```

This function call removes those terms which have at least a 40 percentage of sparse (i.e., terms occurring 0 times in a document) elements.

The TDM can be very sparse, thus we use the method `as.compressed.matrix` of the library `maxent` to store the matrix in the compressed `matrix.csr` format.

Example of terms used as the columns of TDM:

```
[1]  "also"   "april" "bank"    "billion"  "company" "corp"    "cts"
[8]  "dlr"    "dlrs"  "due"     "exchange" "first"   "inc"     "international"
[15] "last"   "march" "market"  "may"      "mln"     "net"     "new"
[22] "one"    "pct"   "reuter"  "said"     "share"   "shares"  "shr"
[29] "stock"  "three" "told"    "two"      "will"    "year"
```

### 4.2.1 Term-document matrix for testing

For test purposes, we have to store original terms used in the learning process, because we have to find their occurences in the test documents. So we have to create another term document matrix with defined columns. This can be done in the folowing way: we computed the TDM as mentioned in the previous paragraph and then we remove all columns that are not in the original TDM matrix:

```
result <- as.DocumentTermMatrix(cbind(newMatrix[,which(colnames(newMatrix)
%in% colnames(origMatrix))],tmpMatrix),weighting=weighting)
```

## 4.3 Binary classifiers ensemble training

The binary classifiers are trained using following libraries: `e1071` for NB and SVM and `randomForest` for randomForest method. Each binary classifier is trained on the same TDM matrix and the ensemble is represented by a list of classifiers.

## 4.4 Voting & decision tree

The output of the ensemble for a particular text category (EARN, ACQ, MONEY-FX) is summed and if the value is greater then 2, the text category is assigned to the given text document. The final target label is determined by concatenating these assigned categories.

For the decision tree, we use the library `rpart`. In the training process, we take the binary output of the ensemble classifiers and create a decision tree that directly assign the target category. We use 70 % of instances for training the binary classifiers and 30 % of instances for training of the decision tree.

# 5 Experiments and evaluation

## 5.1 Experiments

In first experiments, we have trained the ensemble of binary classifiers. The experiments are initially drawn on smaller data set with no parameter tuning. Then we have tried to set up the parameters to increase the performance and finally, we have performed the 5-fold crossvalidation.

### 5.1.1 Learning parameters

Because the classifier training process could be time-consuming on huge training set, the parametes of the binary classifiers have been tuned on smaller data set (5000 documents). It turned out that the baseline accuracy of classifiers is relatively high and the parameter tuning have no significant impact on the accuracy.

| ML method | Parameter | Value |
|---|---|---|
| **SVM** | cost | 300 |
| **RF** | ntree | 500 for category MONEY-FX, 200 for other categories |

The list of learning parameters

The most important parameters are the parameters of creation of the term-document matrix, which were set as shown in the following table:

| Parameter | Value |
|---|---|
| **Stem** | true |
| **Weighting** | Term frequency – inverse document frequency |
| **Sparse** | 0.98 |

## 5.2 Evaluation

For the evaluation of binary classifiers, we have used following performace metrics:

| Metric | Formula |
|---|---|
| **Accuracy** | $\frac{TP + TN}{TP + TN + FN + FP}$ |
| **Precision** | $\frac{TP}{TP + FP}$ |
| **Recall** | $\frac{TP}{TP + FN}$ |
| **F-measure** | $\frac{2 * Precision * Recall}{Precision + Recall}$ |

Where TP means true positive, TN true negative, FN false negative, FP false positive instances.

## 5.3 Binary classifier evaluation

The binary classifiers evaluation using 5-fold cross validation is shown in the following table:

| Classifier | Text Category | Accuracy | Precision | Recall | F-measure |
|---|---|---|---|---|---|
| **SVM** | EARN | 0.9740508 | 0.9290037 | 0.9496512 | 0.9391728 |
| | ACQ | 0.9490624 | 0.7464374 | 0.8756654 | 0.8057058 |
| | MONEY-FX | 0.9808239 | 0.7278682 | 0.8240421 | 0.7726801 |
| **RF** | EARN | 0.9747619 | 0.9483746 | 0.9313143 | 0.9397214 |
| | ACQ | 0.9562365 | 0.8489979 | 0.7768948 | 0.8112193 |
| | MONEY-FX | 0.9854449 | 0.9309424 | 0.6862000 | 0.7890224 |
| **NB** | EARN | 0.7631424 | 0.4664369 | 0.8380253 | 0.5991958 |
| | ACQ | 0.7306981 | 0.2825586 | 0.7948315 | 0.4168544 |
| | MONEY-FX | 0.69572941 | 0.09514365 | 0.78673155 | 0.16966013 |

Binary classifiers 5-fold mean

| Classifier | Text Category | 95 % conf. interval of F-measure mean |
|---|---|---|
| **SVM** | EARN | 0.9390999 - 0.9392456 |
| | ACQ | 0.8054914 - 0.8059203 |
| | MONEY-FX | 0.7714580 - 0.7739021 |
| **RF** | EARN | 0.9396566 - 0.9397861 |
| | ACQ | 0.8108813 - 0.8115573 |

| | | |
|---|---|---|
| | MONEY-FX | 0.7888146 - 0.7892303 |
| **NB** | EARN | 0.5990844 - 0.5993072 |
| | ACQ | 0.4165261 - 0.4171827 |
| | MONEY-FX | 0.1693700 - 0.1699503 |

Binary classifiers F-measure confidence intervals

### 5.3.1 Best achieved results of binary classifiers

We achieved the best results with following methods:

| Text category | Classifier | F-measure | 95 % conf. interval of F-measure mean |
|---|---|---|---|
| **EARN** | RF | 0.9397214 | 0.9396566 - 0.9397861 |
| **ACQ** | RF | 0.8112193 | 0.8108813 - 0.8115573 |
| **MONEY-FX** | RF | 0.7890224 | 0.7888146 - 0.7892303 |

Best binary classifiers for each text category

### 5.3.2 Stemming impact

Interesting experiment is how the word stemming can improve the performance, which is shown in the table below:

| Classifier | Text Category | Without stemming | With stemming |
|---|---|---|---|
| **SVM** | EARN | 0.9678353 | 0.9740508 |
| | ACQ | 0.9347072 | 0.9490624 |
| | MONEY-FX | 0.9753024 | 0.9808239 |
| **RF** | EARN | 0.9746064 | 0.9747619 |
| | ACQ | 0.9512080 | 0.9562365 |
| | MONEY-FX | 0.9850299 | 0.9854449 |
| **NB** | EARN | 0.6546847 | 0.7631424 |
| | ACQ | 0.6667027 | 0.7306981 |
| | MONEY-FX | 0.5996128 | 0.6957294 |

Stemming impact on accuracy

## 5.4 Ensemble evaluation

Evaluation of the ensemble classifiers using 3-fold cross-validation is show in the following tables:

| | Accuracy | 95 % conf. interval of Accuracy |
|---|---|---|
| **Voting** | 0.9093767 | 0.9093176 - 0.9094359 |
| **Decision tree** | 0.9088767 | 0.9088344 - 0.9089191 |

Expected accuracy of each method

| | Precision | Recall | F-measure | 95 % conf. interval of F-measure mean |
|---|---|---|---|---|
| **ACQ** | 0.7704301 | 0.8452576 | 0.8060259 | 0.8055891 - 0.8064626 |
| **EARN** | 0.9472786 | 0.9291158 | 0.9380626 | 0.9379196 - 0.9382057 |
| **MONEY-FX** | 0.7849511 | 0.7750386 | 0.7797479 | 0.7771721 - 0.7823238 |
| **EARN & ACQ** | - | - | - | - |
| **No target label** | 0.9374280 | 0.9255048 | 0.9314139 | 0.9313877 - 0.9314401 |

Expected performance of the majority voting method on target labels

| | Precision | Recall | F-measure | 95 % conf. interval of F-measure mean |
|---|---|---|---|---|
| **ACQ** | 0.7941893 | 0.8131361 | 0.8011995 | 0.8009801 - 0.8014189 |
| **EARN** | 0.9443781 | 0.9314942 | 0.9378891 | 0.9378629 - 0.9379154 |
| **MONEY-FX** | 0.7634673 | 0.7507496 | 0.7566472 | 0.7544534 - 0.7588409 |
| **EARN & ACQ** | - | - | - | - |
| **EAEN & MONEY.FX** | - | - | - | - |
| **No target label** | 0.9374280 | 0.9255048 | 0.9314139 | 0.9313877 - 0.9314401 |

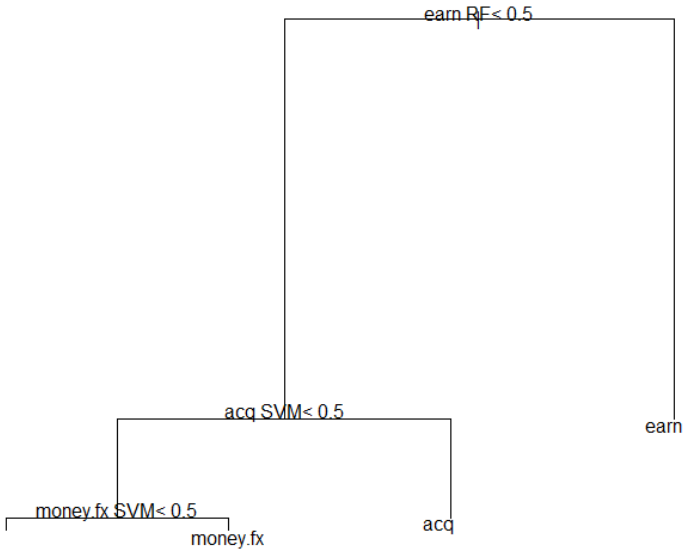
Expected performance of the decision tree method on target labels

We see that the majority voting method has slightly better performace:

| | Accuracy | 95 % conf. interval of Accuracy |
|---|---|---|
| **Voting** | 0.9093767 | 0.9093176 - 0.9094359 |

The best achieved accuracy

In the next picture, we see that the learned decision tree is quite simple and it takes only the input of the SVM and RF binary classifier. There is no leaf for the combination of text categories, thus such decision tree is not able to recognize the document with multiple basic text categories.




## 5.5 Evaluation on test set

### 5.5.1 Evaluation of best binary classifiers

We have evaluated the best binary classifier for each text category on test data set. The results are shown in the following table:

| Text category | Method | Accuracy | Precision | Recall | F-measure |
|---|---|---|---|---|---|
| **ACQ** | RF | 0.9571752 | 0.8197183 | 0.8094576 | 0.8145556 |
| **EARN** | RF | 0.9132191 | 0.6787800 | 0.9613971 | 0.7957398 |
| **MONEY-FX** | RF | 0.9812540 | 0.7424242 | 0.5444444 | 0.6282051 |

Best binary classifiers performance on test set

### 5.5.2 Evaluation of ensemble methods

In the following table we see the result of the voting classifier and ensemble classifier trained on the all training data and evaluated on the test data:

| | **Accuracy** |
|---|---|
| **Voting** | 0.8408209 |
| **Decision tree** | 0.8403361 |

Accuracy of ensemble methods on test set

| | **Precision** | **Recall** | **F-measure** |
|---|---|---|---|
| **ACQ** | 0.7557716 | 0.8687151 | 0.8083171 |
| **EARN** | 0.6724694 | 0.9612903 | 0.7913505 |
| **MONEY-FX** | 0.5826087 | 0.7444444 | 0.6536585 |
| **EARN & ACQ** | - | - | - |
| **No target label** | 0.9553747 | 0.8097050 | 0.8765289 |

Performance of the majority voting method on target labels

| | **Precision** | **Recall** | **F-measure** |
|---|---|---|---|
| **ACQ** | 0.8166432 | 0.8086592 | 0.8126316 |
| **EARN** | 0.6559950 | 0.9631336 | 0.7804332 |
| **MONEY-FX** | 0.5542636 | 0.7944444 | 0.6529680 |
| **EARN & ACQ** | - | - | - |
| **No target label** | 0.9462514 | 0.8166032 | 0.8766599 |

Performance of the decision tree method on target labels

| | | **expected** | | | | |
|---|---|---|---|---|---|---|
| | | No label | ACQ | EARN | EARN & ACQ | MONEY-FX |
| **predicted** | No label | 3403 | 78 | 33 | 2 | 46 |
| | ACQ | 99 | 622 | 3 | 0 | 0 |
| | ACQ& MONEY-FX | 3 | 5 | 0 | 0 | 0 |
| | EARN | 499 | 8 | 1043 | 1 | 0 |
| | EARN & ACQ | 2 | 3 | 2 | 0 | 0 |
| | EARN & MONEY-FX | 2 | 0 | 4 | 0 | 0 |
| | MONEY-FX | 96 | 0 | 0 | 0 | 134 |

Confusion matrix for the majority voting method

| | | **expected** | | | | |
|---|---|---|---|---|---|---|
| | | No label | ACQ | EARN | EARN & ACQ | MONEY-FX |
| **predicted** | No label | 3433 | 120 | 39 | 2 | 37 |
| | ACQ | 128 | 577 | 1 | 0 | 0 |
| | EARN | 530 | 17 | 1045 | 1 | 0 |
| | EARN & ACQ | 0 | 0 | 0 | 0 | 0 |
| | MONEY-FX | 113 | 2 | 0 | 0 | 143 |

Confusion matrix for the decision tree method

# 6 Conclusion

We have seen that from three used machine learning methods, the SVM and RF shows very good performances (f-measure about 93 % for ACQ, 80 % for EARN and 78 % for MONEY-FX) and the NB classifier has poorer accuracy which is near the baseline accuracy (f-measure about 60 % for ACQ, 42% for EARN and 17 % for MONEY-FX). The performance of each method is similar for every text category. We have also seen that the stemming has positive effect on the performance, mainly for the NB classifier. The best binary classifier is RF for each text category.

The performance of the trained classifiers is slightly worse on the test data. For the best binary classifier (RF), the f-measure is about 81 % for ACQ, 80 % for EARN and 63 % for MONEY-FX. For the two ensemble methods, the both have similar results: f-measure about 81% for ACQ, 78% for EARN and 65% for MONEY-FX and the majority voting method has slightly better accuracy. We see that the text categories with more occurrences such as ACQ and EARN are better learnable then the categories with little occurrences as MONEY-FX.

We can conclude that we are able to categorize the documents into the three main text categories, but we have too little learning examples to be able to recognize their combination. Further development could include improving the decision tree ensemble predictor by designing new possibly useful features.