# Information Retrieval.

# “MongoDB. Map-Reduce.”

Students:
Maria Slanova

DMKM, 2016

*In this work we used Python tool for working with MongoDB - Pymongo, Python language processing tool - NLTK, for mathematical functions - Math. Other libraries, that were used - os, glob, re , codecs.*

## Collections

In the very beginning we splitted the corpus (120 text files) in two parts: one for training the classifier and another one for testing. Thus, two folders were created:

Train: *fortnow1 - fortnow30, random1 -random30*
Test: *fortnow31 - fortnow60, random31 - random60.*

To populate the database we presented each document in a following format:

{'content': [content_of_the_file], 'classX': 0/1, 'classY' : 0/1}

To get the file content, we created 2 functions: *create_content* and *create_preprocessed_content.* They both take as an input the filename, and produce as an output the list of words, that a text file contains. The difference is, that the second functions preprocesses the text - it removes all the non-letter symbols and removes the stop-words.
Both keys 'classX' and 'classY' can be either 0 or 1 (their sum is always one). If the filename contains string *"fortnow",* then the file is assigned to class X ('classX':1,'classY':0 ), else to class Y ('classX':0, 'classY':1).
We inserted documents one-by-one using the *insert_one* method. This method automatically adds the id_field to the document and sets the field's value to a generated ObjectId (we don't include id field in the beginning). Also, the method automatically creates the collection: *db.train.insert_one(d)* will create the collection *'train'*.

## Map-Reduce

In Order to perform the computations for Multinomial Naive Bayes classifier we need as input, the aggregated data, in the form of summation for some specific parameters. For this we have constructed 2 sets of Map-Reduce. The first set gives us for each word in the train data set as Key and its count in classX and classY as values, respectively.
(key: word1 , value: [classX: countX , classY: countY] )

The second Map-Reduce set is used to compute the total number of word occurrences in each class for Train data set and the total number of words in the train collection.
(key: classX , value: totalCountX)*
(key: classY , value: totalCountY)
(key: TotalCount , value: totalCount)

After we have the above summated values from Map-Reduce, we can perform training for Multinomial Naive Bayes classifier.
Once we trained the model with the formulas provided in the lab. We need to compute the prediction of a new document using the parameters *n* the total number of documents in the corpus, n(cj) the number of documents of a given class, count(cj) the number of word occurrences within documents of each class, and count(wi; cj) the number of occurrences of each word within documents of each class. Having these values from Map-Reduce, we can compute the prediction of a new document using the formula given in lab document.

**Results on initial data**

In the following we present the results of classifying the test data set, which wasn't preprocessed.

| | fortnow | random |
|---|---|---|
| fortnow | 19 | 11 |
| random | 1 | 29 |

If we denote *a* - number of fortnow documents, classified as fortnow; *b* - number of fortnow documents, classified as random; *c* - number of random documents, classified as fortnow; *d* - number of random documents, classified as random, then we can compute:

$$Precision = \frac{a}{a+c} = \frac{19}{20} = 0.95\ (95\%)$$
$$Recall = \frac{a}{a+b} = \frac{19}{30} = 0.63\ (63\%)$$
$$Accuracy = \frac{a+d}{60} = \frac{18+29}{60} = 0.8\ (80\%)$$
$$F1-score = \frac{2*Precision*Recall}{Precision+Recall} = 0.76\ (76\%)$$

We can see that the error rate of our classifier is pretty high i.e. 22%

## Results on preprocessed data

Below are shown the results of classifying the test data set, which was initially preprocessed.

| | fortnow | random |
|---|---|---|
| fortnow | 26 | 4 |
| random | 3 | 27 |

$$Precision = \frac{a}{a+c} = \frac{26}{29} = 0.9\ (90\%)$$
$$Recall = \frac{a}{a+b} = \frac{26}{30} = 0.87\ (87\%)$$
$$Accuracy = \frac{a+d}{60} = \frac{26+27}{60} = 0.88\ (88\%)$$
$$F1 - score = \frac{2*Precision*Recall}{Precision+Recall} = 0.88\ (88\%)$$

The results on preprocessed data are much better - the error rate is smaller than in previous case i.e. 12%.

## Conclusions

In this work we perform Multinomial Naive Bayes classifier using Map-Reduce for text categorization task. We conduct the experiment on two types of data i.e *unprocessed* and *preprocessed*. We can see that the classifier performed well with 12% of error-rate for the preprocessed data, whereas for unprocessed data we have 22% of error-rate. The experimental evaluation allows us to draw conclusions about the promise, that the preprocessing transformations lead us to improved results with some extra computational overhead.