Documents classifier

Tomasz Kulik

2019

Next chapter

Who am I

Tomasz Kulik

- Software Developer in Nokia
- MSc in Computer Science, graduated from University of Science and Technology in Wrocław
- Interested in programming, algorithms, machine learning

Scope of the presentation

Documents classifier

- Information about the dataset
- Text preparation
- TF-IDF method desciption
- Reccurent Neural Network method desciption
- Application

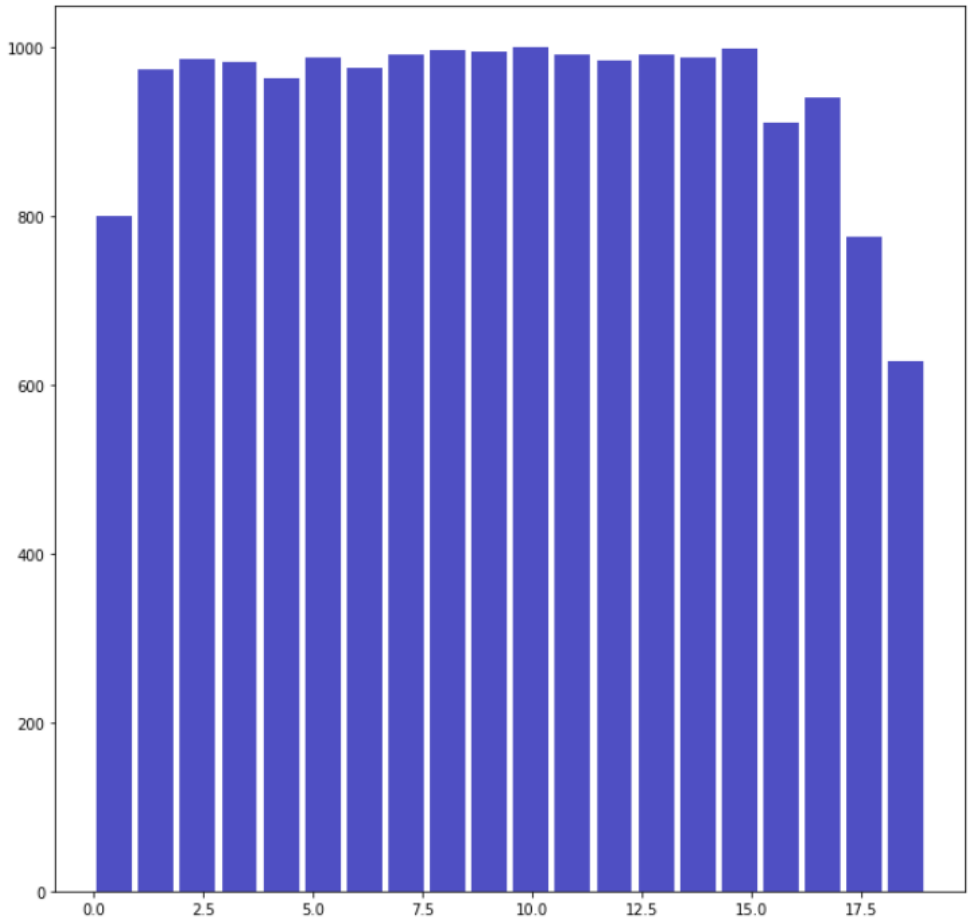
Next chapter

Dataset

- 18846 documents
- 20 subjects
- 83% of data used as training set

Dataset

Distribution of the dataset:



Tools

- Python 3 - Keras, SciKit learn, NLTK, (...)
- Jupyter notebook

Next chapter

Stop words

- Ignore words that can be used in every context (most likely meaningless in problem of classification)
- In english for e.g. (*'ourselves'*, *'hers'*, *'between'*, *'yourself'*, *'but'*, *'again'*, ...)

Stemming

In linguistic morphology and information retrieval, stemming is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form—generally a written word form.
Human readable:

- Keep only the root of the word, despite the form used in the processed text.

Stemming

Example:

- Going, goes, gone $\rightarrow$ go
- Went $\rightarrow$ went
- Change, changing $\rightarrow$ chang

Lemmatization

Lemmatization in linguistics is the process of grouping together the inflected forms of a word so they can be analysed as a single item, identified by the word's lemma, or dictionary form.
Human readable:

- Use a basic word (dictionary form) instead of inflected one.
- It helps to track the basic meaning of a processed text (not the whole context/meaning).

Lemmatization

Example:

- Go, goes, went $\rightarrow$ go
- Buy, bought, buying $\rightarrow$ buy

Stemming/Lemmatization

- Lower dimensionality (less different words to focus on)
- Better generalization effect in case of categorization

Term Frequency - Inverse Document Frequency - (*TF-IDF*)

Numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.

Term Frequency - (TF)

For every document calculate a percentage occurence of every word.

$$tf(t,d) = \frac{freq_{t,d}}{N_d}$$

	go	start	exam	(word)	(...)
Doc1	0.3	0.14	0.12	0.07	...
Doc2	0.2	0.4	0.02	0.01	...
...					

Inverse Document Frequency - (*IDF*)

Calculate the importance of each word in document by computing the measure:

$$idf(t, D) = \frac{|D|}{|\{d \in D : t \in d\}|}$$

The higher is the IDF coefficient, the more "original" is the word.

Term Frequency - Inverse Document Frequency - (*TF-IDF*)

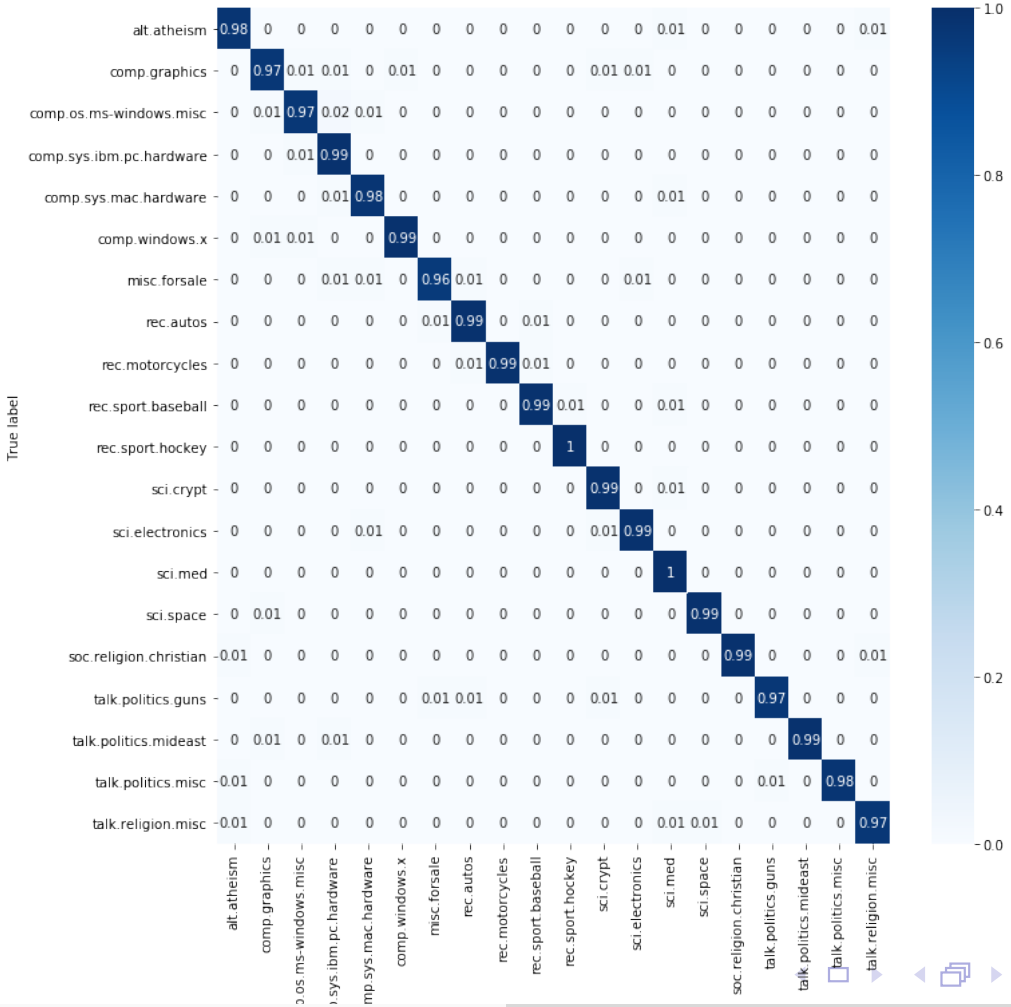
$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D)$$

With equation above we can compute the TF-IDF measure for each word in every document.
Result:

$$\text{One document} = \text{one vector}$$

Results after training

Accuracy: **93,5%**, F1-Score (weighted): **98,4%**



Next chapter

Embedding

Popular method in recommendation algorithms (e.g. YouTube)
Words or phrases from the vocabulary are mapped to vectors of real numbers. E. g.

$$\text{'word'} = [2.23,\ 33.,\ 0.2,\ ...]$$

Used also as a dimensionality reduction technique.

- $f(w) = [x_1, x_2, ..., x_n]$
- Problem: Minimize a distance between similar/interchangeable words in n-dimensional space (and increase distance between unconnected words).

Convolutional Neural Network

The Conv Net is a composition of two types of layers:

- Convolutional layers
- Pooling layers

Convolutional NN are able to automated feature extraction from input data (in case of text for e.g. sentece order, words co-occurence; in image processing edges, parts of face etc.)

Recurrent Neural Network

- Recurrent cells instead of simple neurons (e.g. Long Short Term Memory).
- Hidden state handled between timesteps processing.

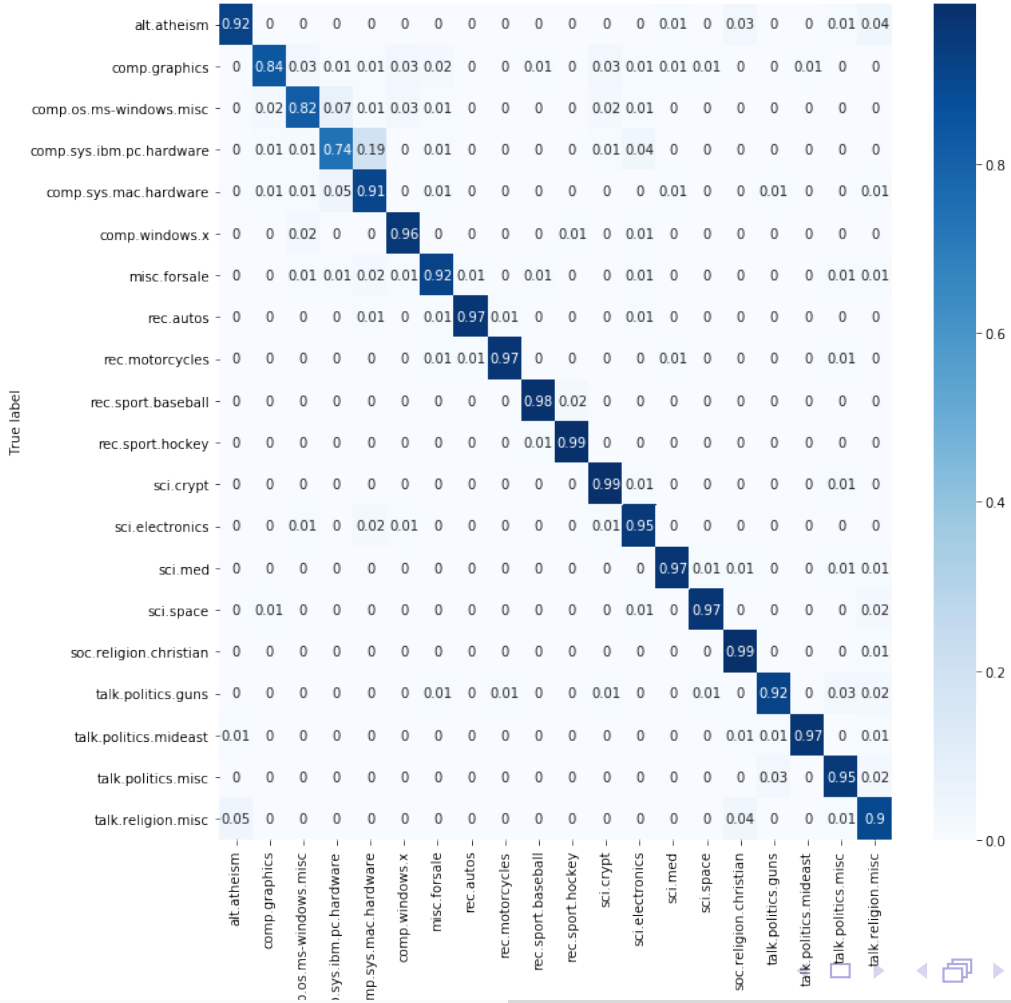
RNN can ,,remember" the context of the text, for instance a gender of a subject.
They are good in predicting time series and generating text based on input words (text generators, machine translators, etc.).

Knowledge transfer

- Use pre-trained embedding layer as an input for classifier.
- Pre-trained Conv layers as input for further layers in Deep NN (popular in image processing).

Results after training

Accuracy: **80,3%**, F1-Score (weighted): **93,3%**

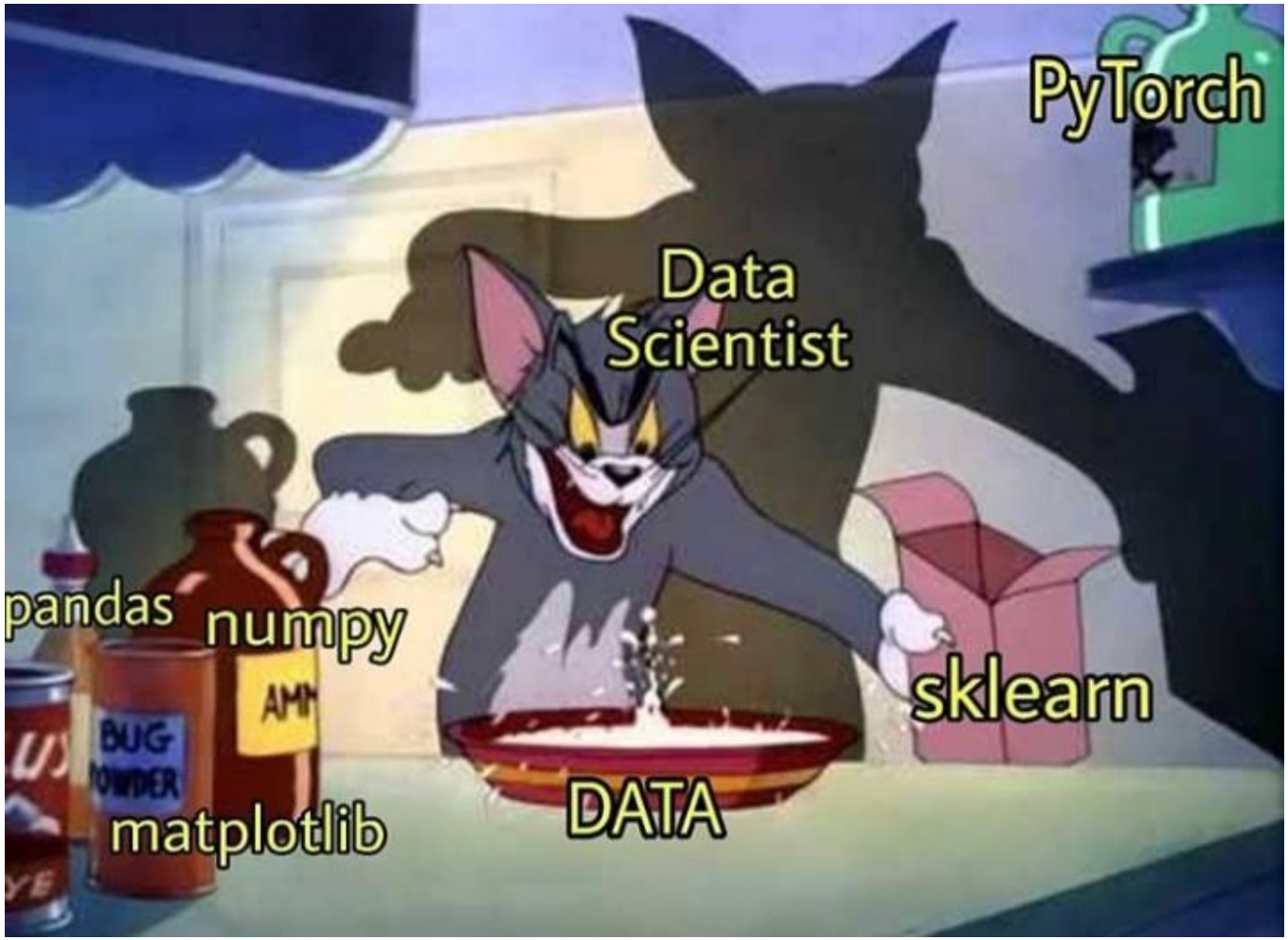


Next chapter

Application

- Script written in Python3
- Model and parameters exported from Jupiter to external files and loaded during script execution
- Simple user interface: *./classifier.py path/to/article.txt*
- Result printed in terminal after few seconds
- Further steps:
  - Run script as a service to prevent loading model per each document.
  - Implement REST API to let other services communicate with the script on demand.

Thank you!


PyTorch
Data Scientist
pandas
numpy
sklearn
matplotlib
DATA