Classification of biochemical texts using statistical learning methods

An Overview

Sebastian Schmeier

- Motivation
- Background
- Aim
- Proceeding

- **Motivation**
- Background
- Aim
- Proceeding

In Silico Modelling

Modelling of biological Systems

in silico:

- Mathematical Models
- Kinetics (Michaelis-Menten, Hill,…)

➔ *Kinetic Data*

Kinetic Data

Constants

(e.g. Michaelis-Menten-Constant, Hill-Coefficient, kcat, vmax)

This data is measured in elaborate experiments, that we can‘t do ourselves

➔ money, time, employees

Literature Research

- Searching such kinetic data in existing literature

  (here: Online-Journals)
- Only few data in a huge literature diversity
- To make a decision with regard to the relevance of a particular article

➔ **Automatic Process?**

- Motivation
- **Background**
- Aim
- Proceeding

Already done

- Python Program
  - Load article randomized (PDF)
  - 13 Online-Journals
  - PDF ➔ Text
  - Simple full text keyword search

The Data Set

- Loaded and searched ***5000*** articles
- Found relevant keywords in ***900*** articles
- After 3 months of reading:
  - **100** articles ➔ good
  - **800** articles ➔ bad

- Motivation
- Background
- **Aim**
- Proceeding

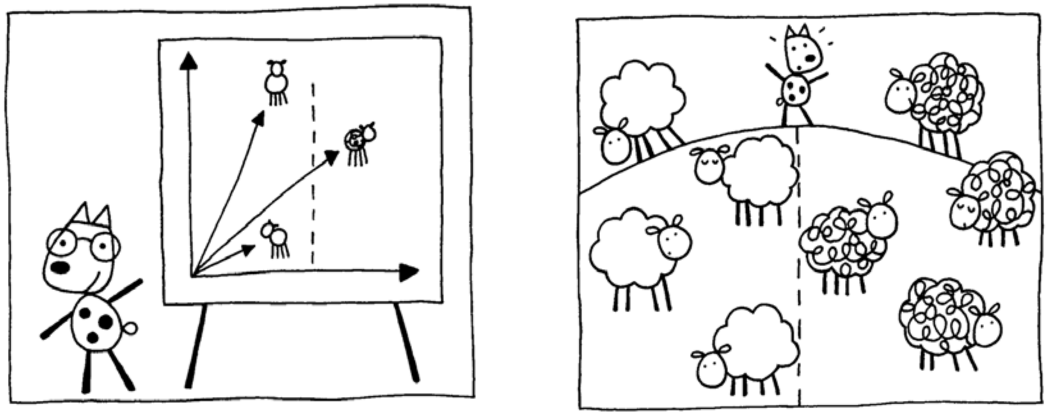
Bachelor-Thesis

- Building on this data set, an algorithm will be implemented, which can hopefully classify any new text with regard to its relevance for kinetic modells.

- Motivation
- Background
- Aim
- **Proceeding**

SVMs

- SVM (support vector machine)
- A statistical learning method to train a classifier, which makes a decision on certain problems.



SVMs

- A feature vector represents an object, that has to be classified.
- Based on vector similarities, which means the distance between them, objects are clustered into different classes.
- Then a new object can be assigned to one of the classes.

Applied to this problem

1. A *word vector* has to be build, which contains all unique words of all 5000 articles.
2. For each of the 900 known articles a *term frequency vector* (tf) has to be build, that contains the absolute number of words in this article.

➔ *feature vector*

Applied to this problem

3. A *document frequency vector* (df) has to be build, that indicates in how many articles a specific word appears.
4. tf´s have to be weigthed

➔ tf*idf

tf = 1 + log($tf_{t,d}$)

idf (inverse document frequency) = log(N / df_t)

NLP

To obtain a *word vector* and the *term frequency vectors,* methods of Natural Language Processing (**NLP**) come into play:

- *Tokenization*
- *Part-of-Speech-Tagging*
- *Stemming*

TreeTagger

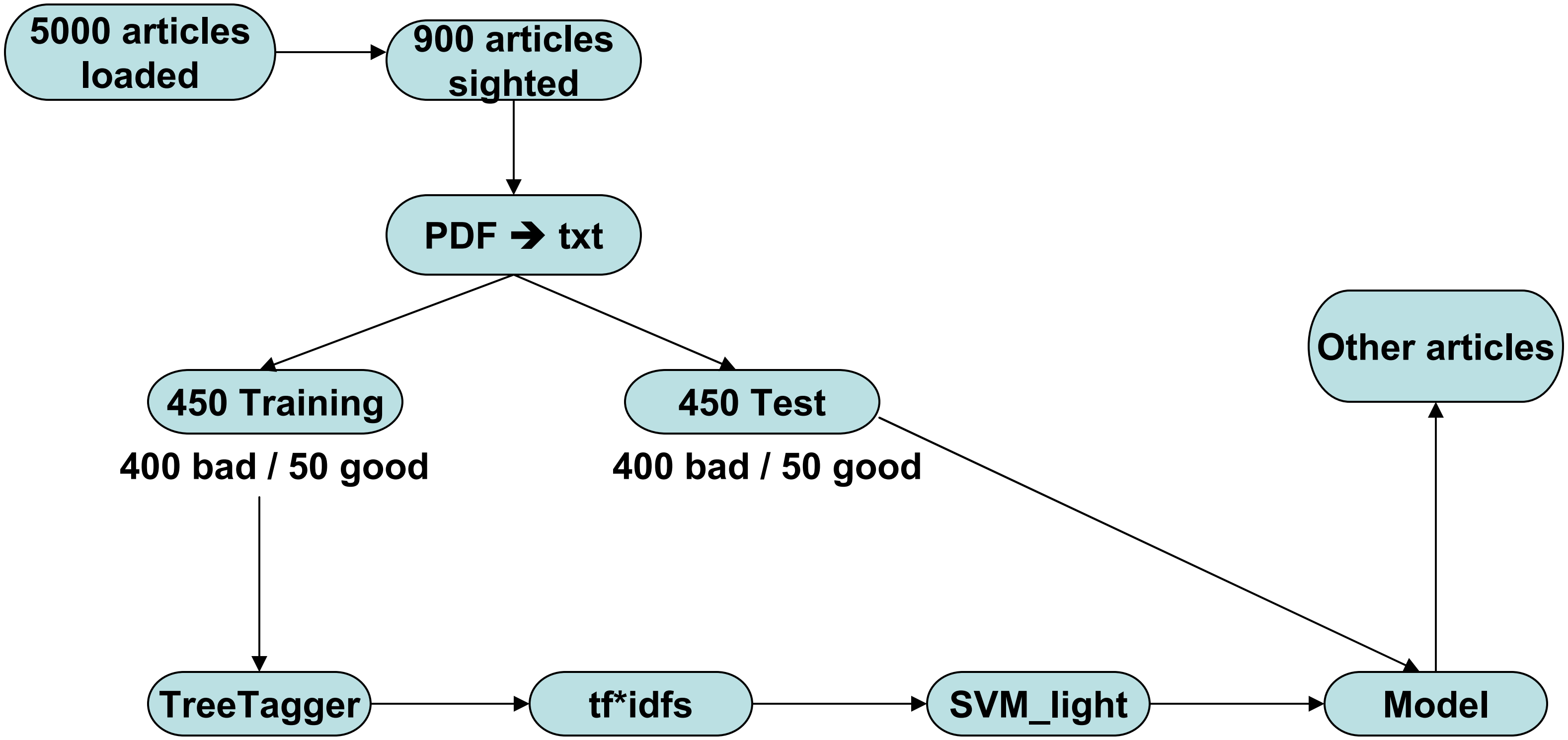
- A tool for annotating text.
- The input is standard text.
- The output:

which	WDT	which
is	VBZ	be
thought	VVN	think
to	TO	to
be	VB	be
the	DT	the
greatest	JJS	great

Institute for Computational Linguistics of the University of Stuttgart

http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html

Overview


5000 articles loaded
900 articles sighted
PDF ➔ txt
450 Training
400 bad / 50 good
450 Test
400 bad / 50 good
Other articles
TreeTagger
tf*idfs
SVM_light
Model