Projektgruppe

Information-Driven Software Engineering

Steffen Beringer

Categorization of text documents via classification

4. Juni 2010

Content

- Motivation
- Text categorization
- Classification in the machine learning
- Document indexing
- Construction methods
- Evaluation

Motivation

- Document organization (newspapers)

Topic ?

Politics
Home News
Economy
Lifestyle
Cars
...

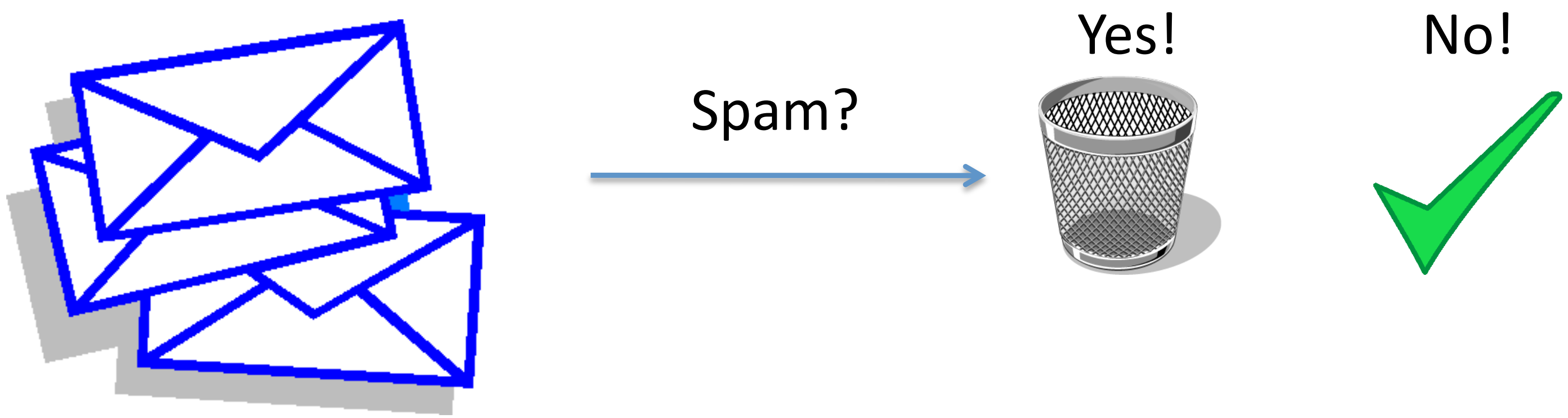
Place ?

Africa
America
Asia
Europe
Middle East
...

...

Motivation

- Text filtering (E-mail filter)



Motivation

- Word Sense Disambiguation / resolving natural language ambiguities

„Bank“ —Sense?→

Motivation

iPhoneFan99, 1.1.2010:
Meine Meinung: Das iPhone ist toll! (...)

Topic?

iPhone, iPod, iPad

documents D | function f | categories C

$d \in D$ | f | $c \in C$

Text categorization: A Definition

documents D function f categories C

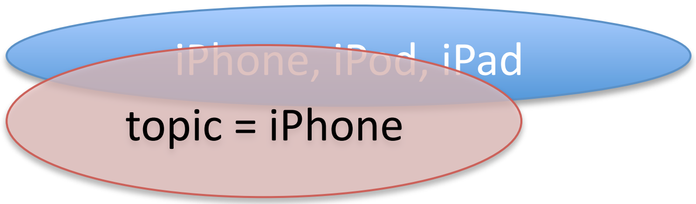
→Goal: approximate the unknown target function f: D → C

- Properties:
  - Just symbolic labels (no „meaning“ of labels)
  - No exogenous knowledge
- Different constraints

Text categorization: Single- vs. Multilabel

iPhoneFan99, 1.1.2010: Meine Meinung: Das iPhone ist toll! (...) Das iPad ist doof!

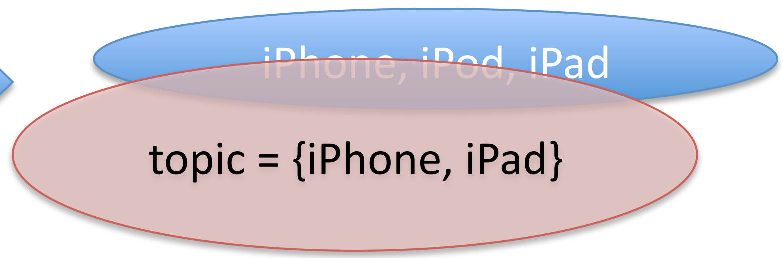
$\xrightarrow{\text{topic}}$



$$f : D \rightarrow C$$

iPhoneFan99, 1.1.2010: Meine Meinung: Das iPhone ist toll! (...) Das iPad ist doof!

$\xrightarrow{\text{topic}}$



$$f : D \rightarrow Pow(C)$$

Text Categorization: „Hard“ vs. Ranking

iPhoneFan99, 1.1.2010: Meine Meinung: Das iPhone ist toll! (...) Aber es kann kein Multitasking

mood →

Positive, Negative

Positive = True
Negative = False

$$f: D \times C \rightarrow \{True, False\}$$

iPhoneFan99, 1.1.2010: Meine Meinung: Das iPhone ist toll! (...) Aber es kann kein Multitasking

mood →

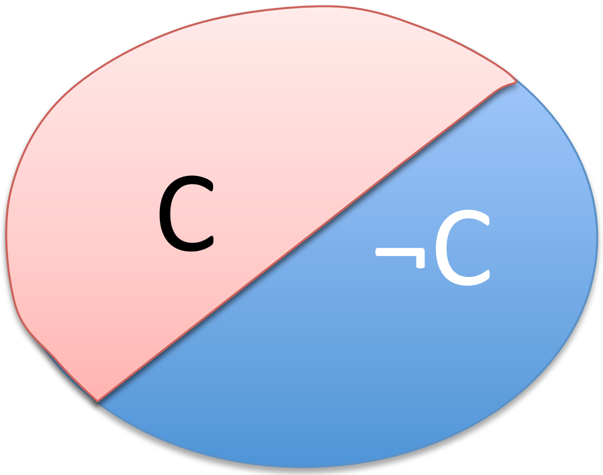
Positive, Negative

Positive = 0.7
Negative = 0.4

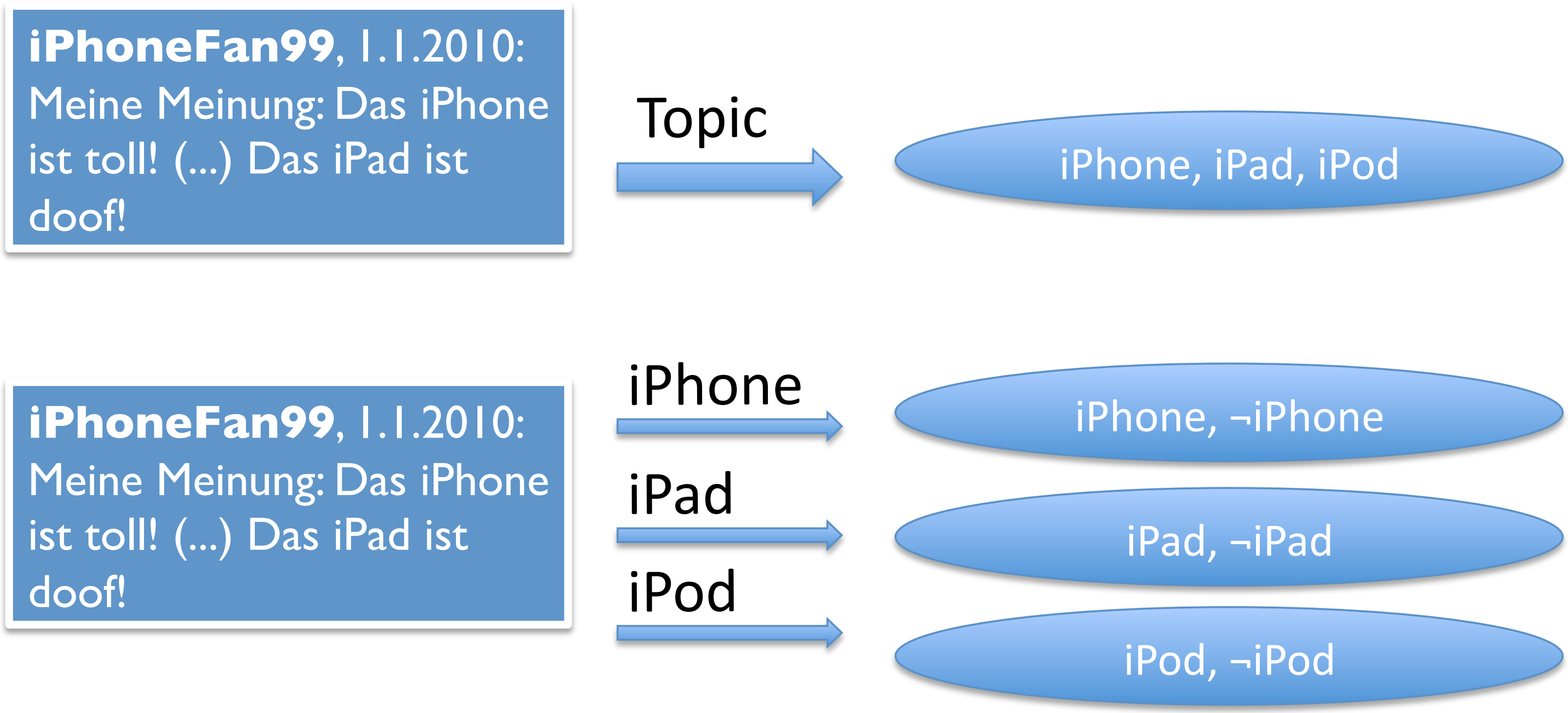
$$f: D \times C \rightarrow R$$

Binary Categorization

- Only two categories C and ¬C
- f: D → {C, ¬C}
- Some classifier only support this type of classification
- Is this a problem?
- Transform multilabel classification with C = $\{c_1,\ldots c_n\}$ into |C| independent problems of binary classification $\{c_i, \neg c_i\}$



Binary Classification



Text classification: The knowledge engineering approach

- In the 80s most popular: knowledge engineering
  - System consisting of a set of **manually** defined logical rules (DNF rules)
  - `If(`*`iPhone & toll`*`) or`

    `(`*`iPhone & ¬schlecht`*`) or`

    `(`*`Touchscreen & Handy`*`)then` **`IPHONE`** `else` **`¬IPHONE`**

  - → knowledge aquisition bottleneck ☹

Text classification: The machine learning approach

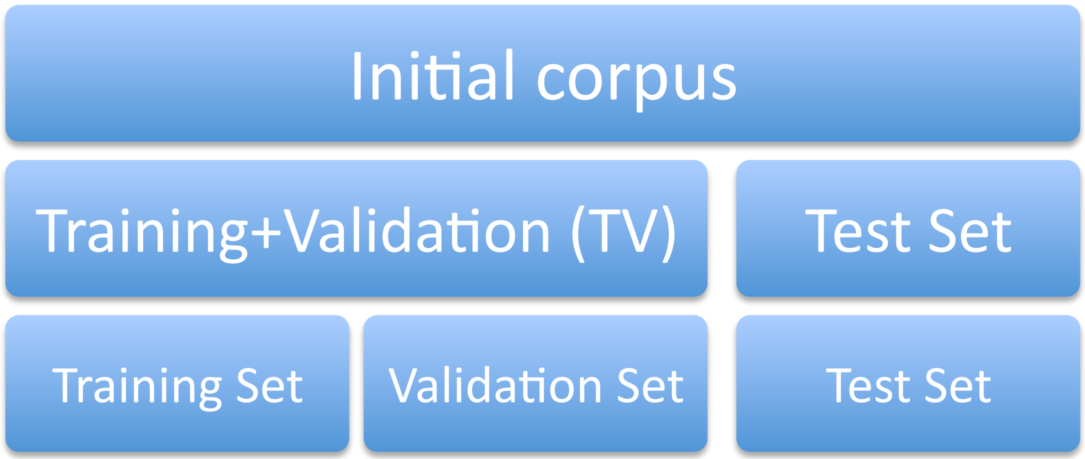
- General inductive process (learner) builds the classifier
- Supervised learning: inductive, **automatic** construction of a classifier from a set of **manually classified documents**
- Preclassified documents are the key resource!

Approach for Automation

1. Classify some real examples manually
2. Transform documents into a representation suitable for learning algorithm and classification task
3. Find relations between features and document class and try to approximate ideal function

Initial corpus: Training Set, Test Set and Validation Set

- Initial corpus: preclassified documents with positive and negative examples



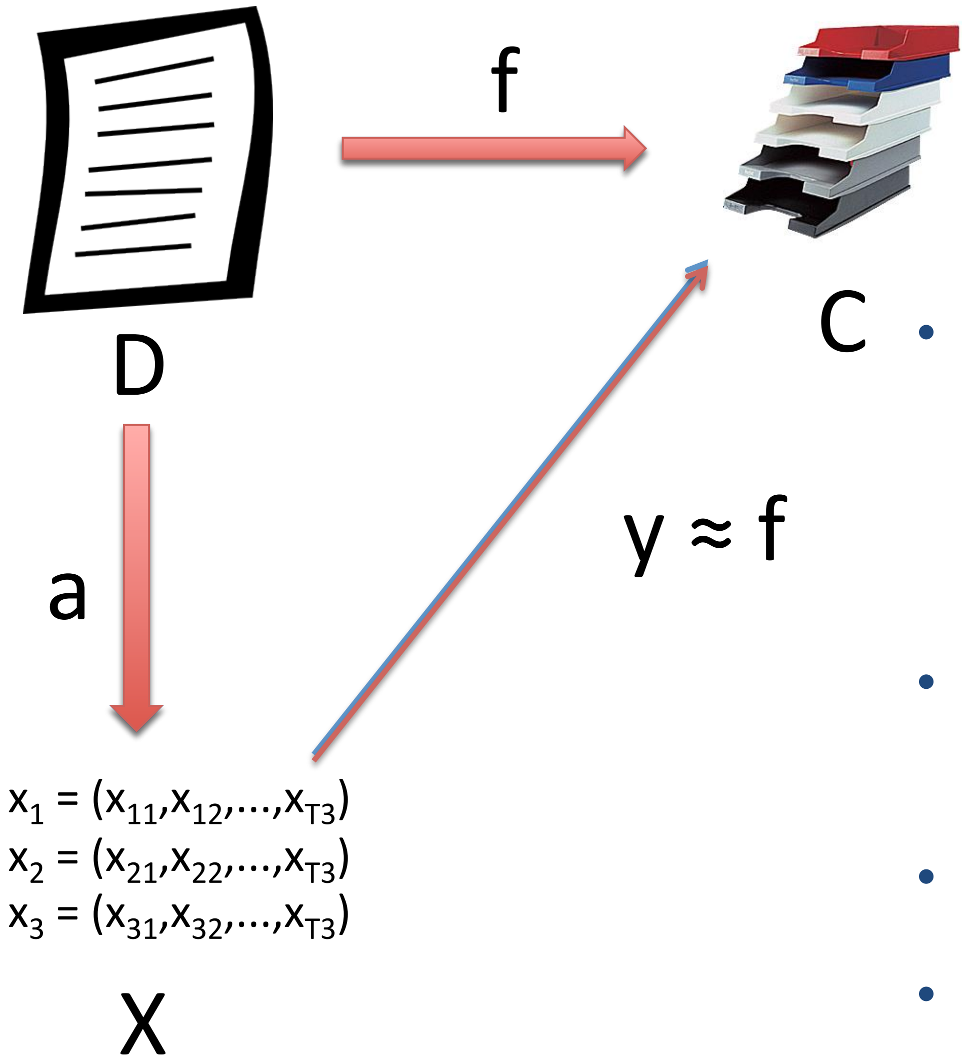
- Training Set → construct classifier
- Validation Set → optimizing, parameter tuning
- Test Set → testing effectiveness

Document indexing

- How to represent documents

Generate initial corpus → Index documents → Construct classifier

Document indexing



Documents typically strings of characters, cannot be direcly interpreted by classifier

- Transformed into a representation suitable for learning algorithm and classification task
- usually represented as a vector of term weights / features
- $d_j=(w_{1j},..,w_{Tj})$
- Classifier approximates

Document indexing

- Different approaches
  - Different ways to understand what a term is
  - Different ways to compute term weights
- Examples:
  - Set of Words / Bag of words
  - Average Word Frequency Class
  - Part of Speech
  - Genre-Specific Core Vocabularies
  - Gini Coefficient

Set of Words / Bag of words

- Idea: Each distinct word w_i corresponds to a feature with the number of times w_i occurs in the document as its value

iPhoneFan99, 1.1.2010:
Meine Meinung: Das iPhone ist toll! (...)

→

w_0 = 1 (toll)
w_1 = 1 (Meinung)
w_2 = 0
w_3 = 0
…
w_n = 0

- Problems: very big feature vectors
- Optimizations:
  - Word stemming
  - Skip „stop-words“ (and, or, ...)

Average Word Frequency Class

- Idea: measure complexity of language usage
- For each word w in domain D compute word frequency class c(w)
- c(w*)= 0 → w* denote the most frequent word
- Most uncommonly words have frequency class 19
- $c(w) = \lfloor \log 2(f(w*)/f(w)) \rfloor$

Word	Instances	c(w)
The	3789654	0
He	2098762	log2(3789654/2098762) = 0
Boy	56975	log2(3789654/56975) = 6
Outragious	76	log2(3789654/76) = 15
Stringyfy	5	log2(3789654/5) = 19

More features on complexity of language usage: readability analysis

- Automated readability index, Coleman- Liau index
  - Measure characters / words and words / sentences
- Gunning fog index
  - Measure words / sentences and complex words / words
- Flesh-Kinaid readability test
  - Measure words / sentences and syllables / words

Part of Speech

- Idea: label the words of a sentence according to their function or word-class (Nouns, verbs, adjectives, adverbs...)

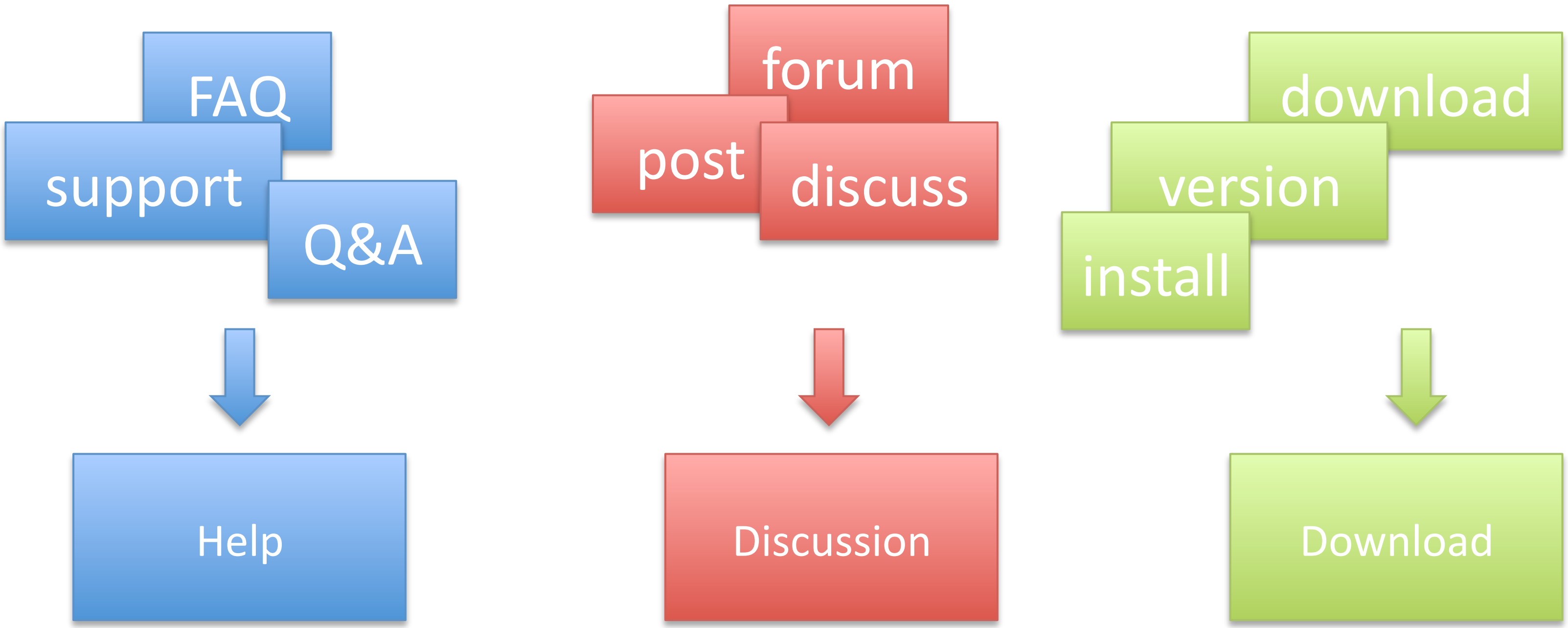
iPhoneFan99, 1.1.2010: Meine Meinung: Das iPhone ist toll! (...)

noun	alphan um.	prono un	noun	article	noun	verb	adjecti ve

More on this topic in the talk of Michael Meier

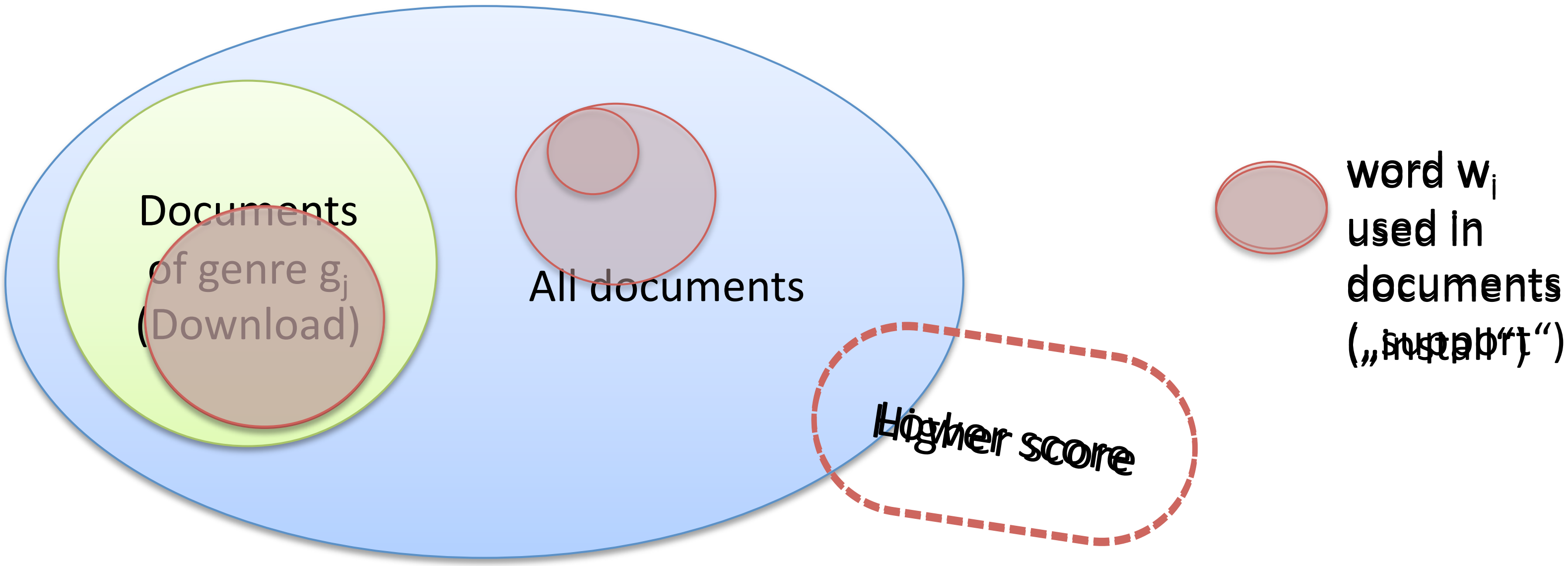
Genre-Specific Core vocabularies

- Idea: compile word sets that may be specific to a certain genre



Genre-Specific Core vocabularies: mining core vocabularies

- What are words characteristic for a specific genre?
  - Frequently used in some genre class g
  - Rarely used in all other classes
- Computation of a score for each word w_i and genre g_j:

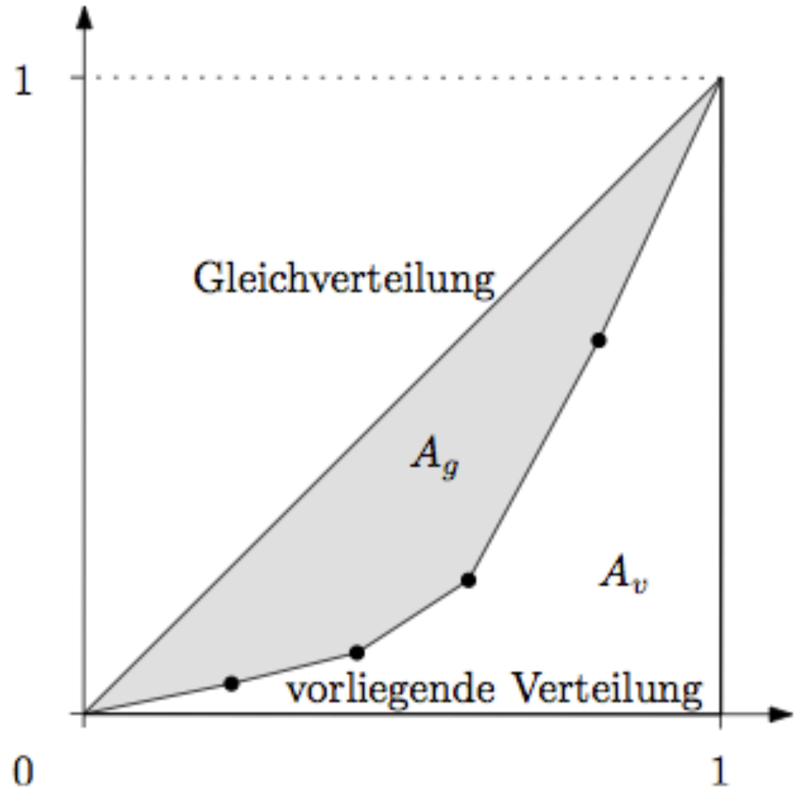


Genre-Specific Core vocabularies: measure core vocabulary

- Ideas to measure presence or absence of core vocabulary:
  - Determine coverage of core vocabulary of document
  - Determine how the core vocabulary is distributed over the document

Gini Coefficient

- Idea: Measure **distribution** of vocabulary in the document
- $G = A_g / (A_g + A_v)$

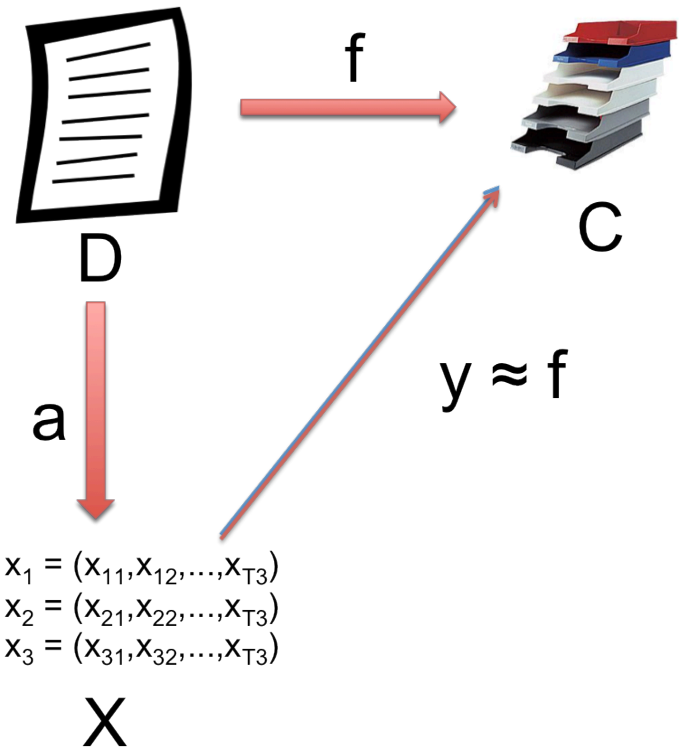


More features

- Syntactic Group Analysis
- Text Statistics
- Presentation Related Features

Inductive Construction of Text Classifiers

- Approximates the classification function f
- Examples:
  - Bayes classifier
  - Decision rule classifier
  - Neuronal networks
  - Example based classifiers
  - Support vector machines



Generate initial corpus → Index documents → **Construct classifier**

Inductive Construction of Text Classifiers: 3 Example Approaches

- (Naive) Bayes Classifier
- Descision Tree Classifier
- Support Vector Machines

Probabilistic Classifier: (Naive) Bayes Classifier

- Idea: compute probability that a document $D = (d_1..d_n)$ belongs to category C: $P(C|D) = P(C \cap D)/P(D)$
- Based on Bayes Theorem: $P(A|B) = (P(B|A)*P(A))/P(B)$
- Assumptions:
  - Binary classification
  - Features independent of each other

Probabilistic Classifier: (Naive) Bayes Classifier

- $P(C|D) = (P(D|C)*P(C)) / P(D)$ (Bayes Theorem)
- We know:
  - $P(C)$ = #documents in category C / #documents
  - $P(d_i|C)$ = #documents in category C containing feature d_i/ #documents in C
  - $P(D|C) = P(d_1|C)*P(d_2|C)*...*(d_n|C)$ (with independence)
- $P(C|D) = (P(d1|C)*..*P(d_n|C)*P(C))/P(D)$

Probabilistic Classifier: (Naive) Bayes Classifier

- Assume binary classification $C = \{S, \neg S\}$
- Compute proportion $Q = P(S,D)/P(\neg S,D) = (P(d_1|S)*..*P(d_n|S) *P(S))/(P(d_1|\neg S)*..*P(d_n|\neg S)*P(\neg S))$

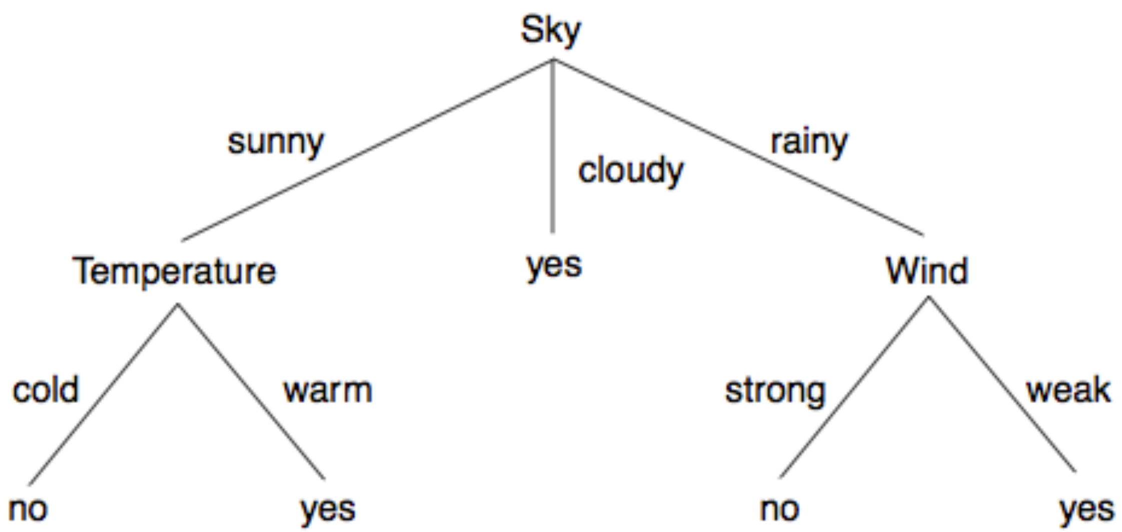
Probabilistic Classifier: (Naive) Bayes Classifier

- Feature properties:
  - All features identically important
  - Statistically independent →mostly not true ☹
- Works fine in practice
- Not easily interpretable by humans

Descision Tree Classifier

- Idea: disjoint decomposition of documents via a tree

Example	Sky	Temperature	Humidity	Wind	Water	Forecast	EnjoySport
1	sunny	warm	normal	strong	warm	same	yes
2	sunny	warm	high	strong	warm	same	yes
3	rainy	cold	high	strong	warm	change	no
4	sunny	warm	high	strong	cool	change	yes

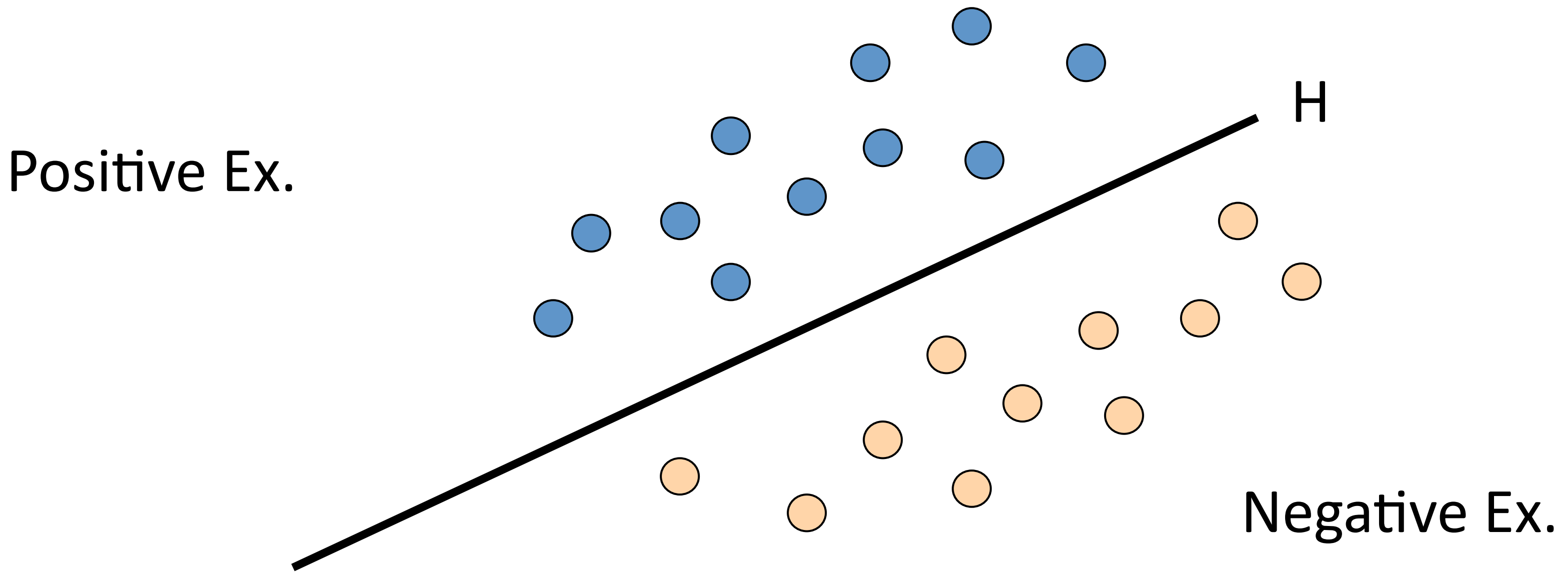


Descision Tree Classifier

- Symbolic / nonnumeric algorithm
- How to construct the tree?

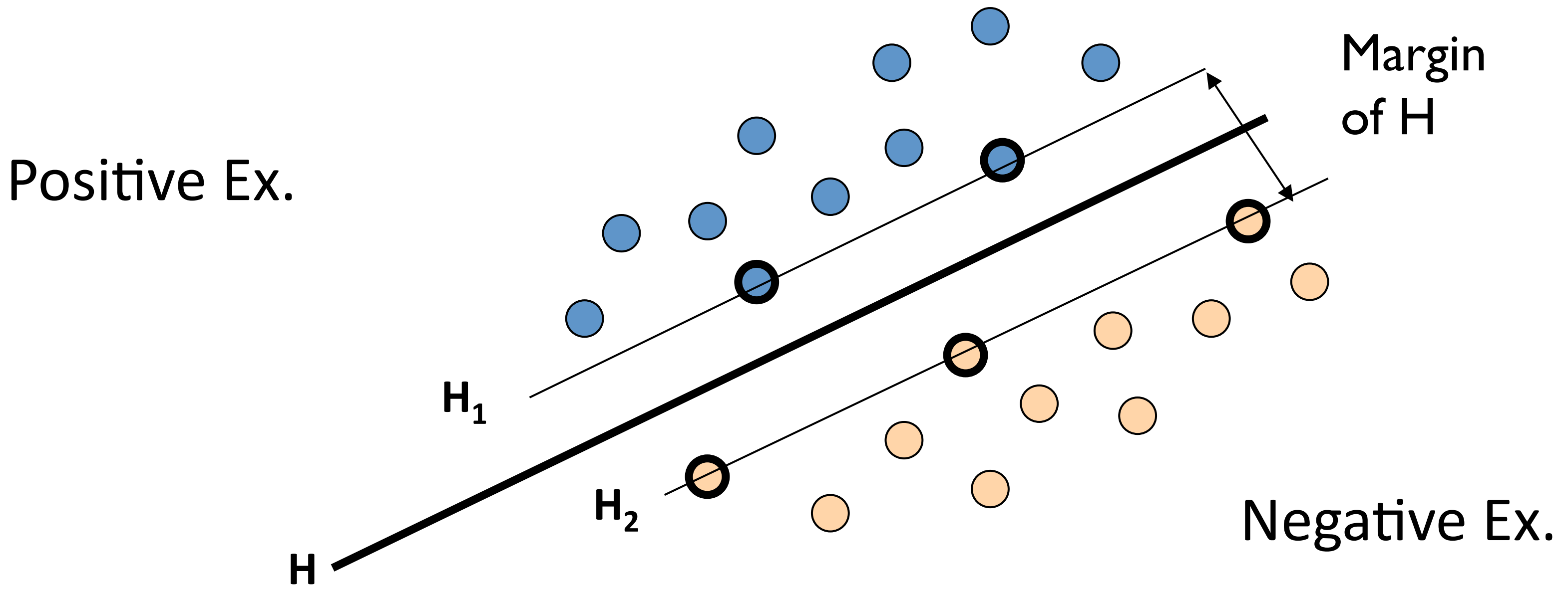
Support Vector Machines

- Idea: Separation of the documents by hyperplane H (desision surface) in the T-dimensional space



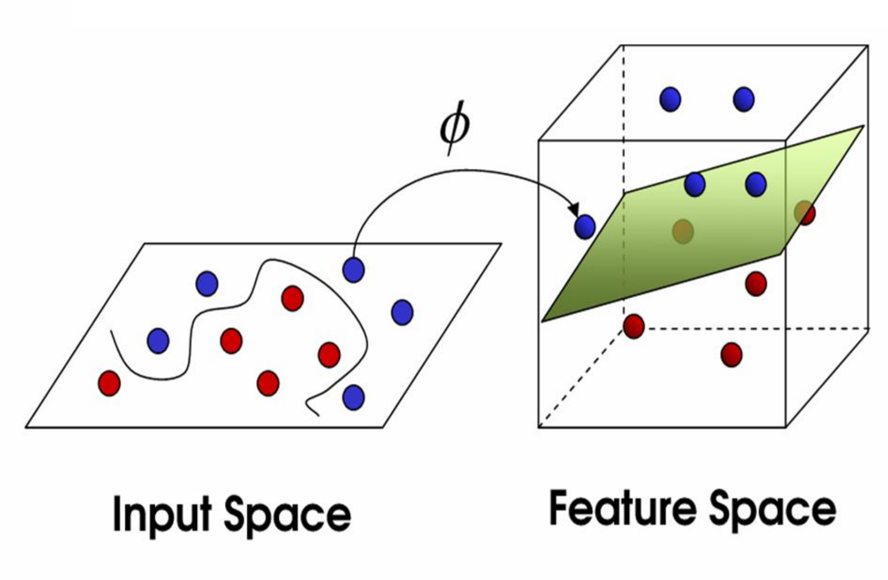
Support Vector Machines

- Construction: maximize the minimum margin of H



Support Vector Machines

- Best understood for binary classification
- Also applicable if positives and negatives are not linearly separable
- Few parameter tuning

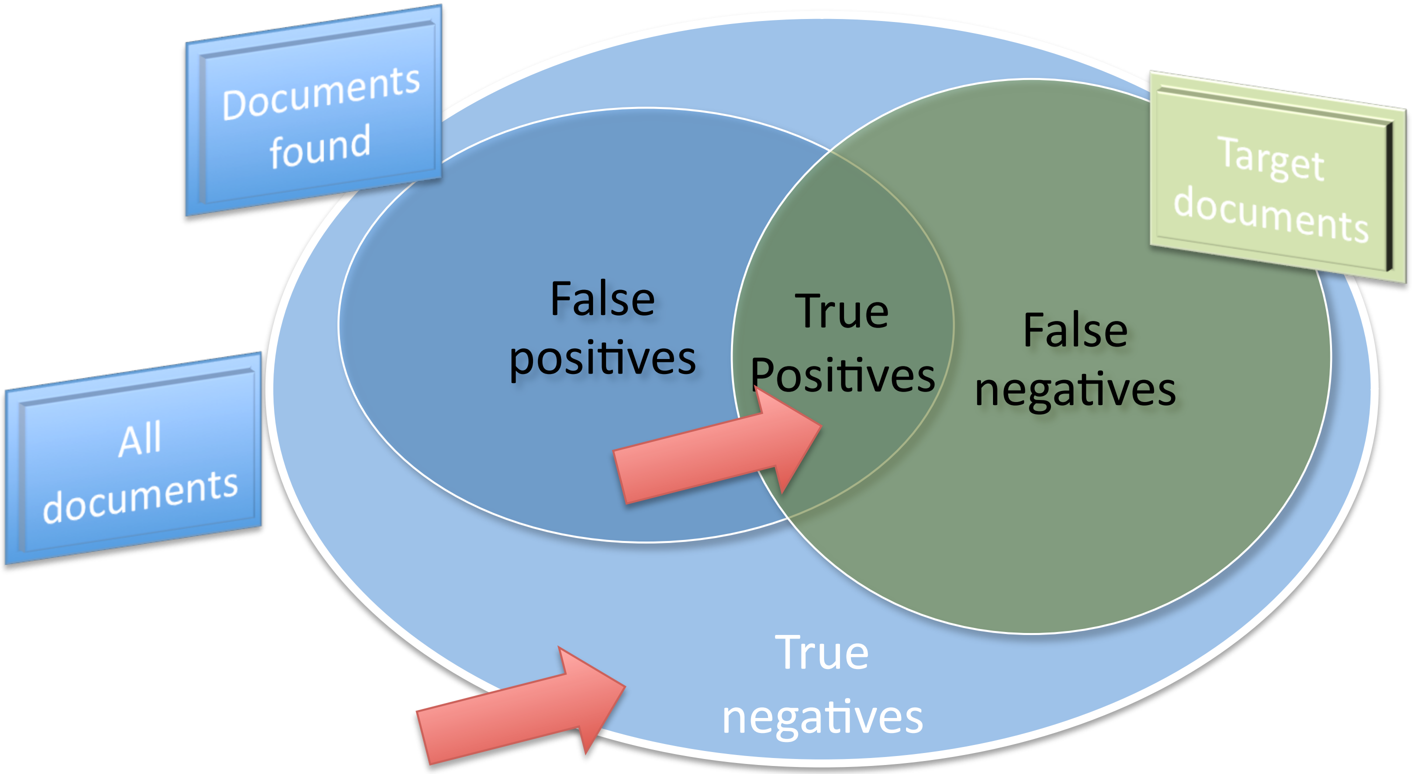


Evaluation

- Typically conducted experimantally
- Usually measure is **effectiveness**, not efficiency
- Measures for effectiveness:
  - Accuracy & Error
  - Precision and Recall
  - Micro- and Macroaveraging

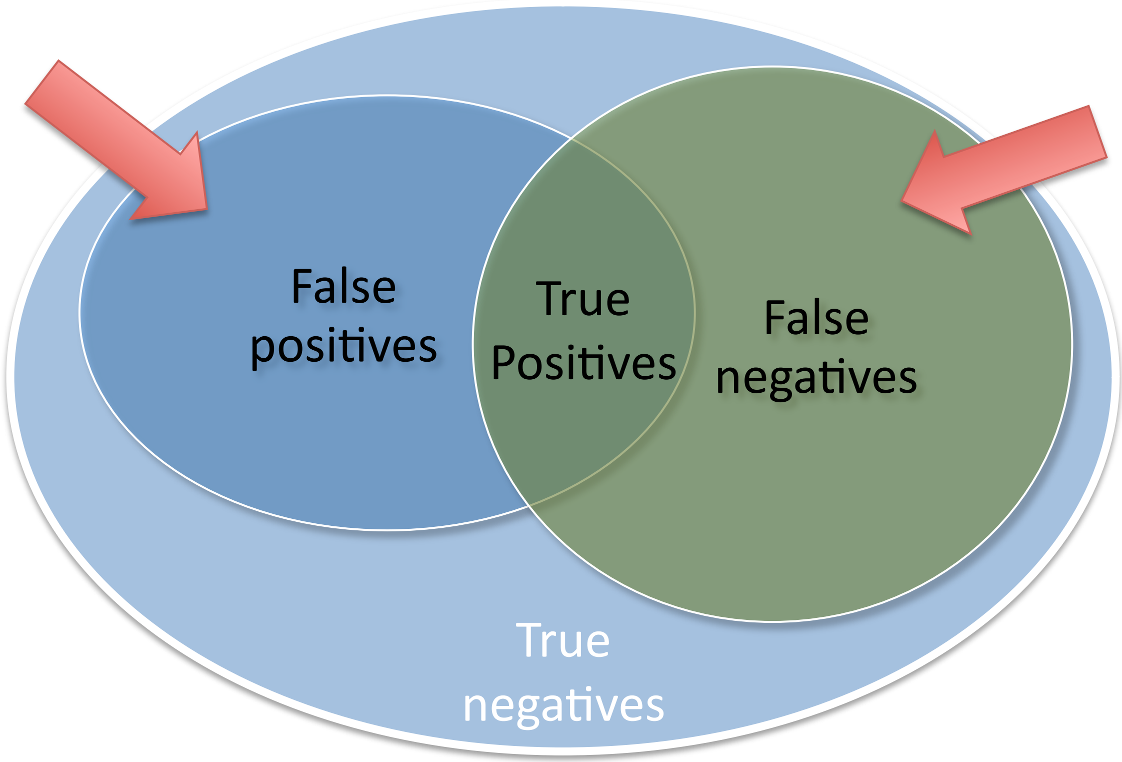
Accuracy

- Accuracy: the fraction of the **correct** classifications
- Accuracy A = (TP+TN)/(TP+TN+FP+FN) = „correct“/|D|



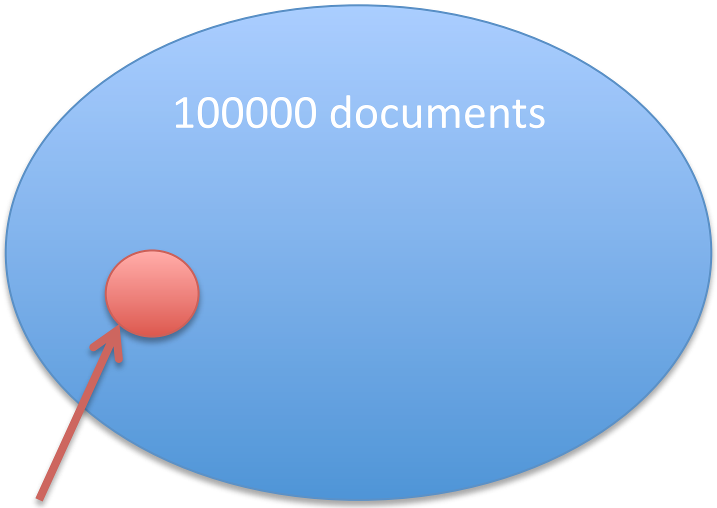
Error

- Error: fraction of the **incorrect** classifications
- Error = (FP+FN)/(TP+TN+FP+FN) = 1-Accuracy



Accuracy and Error

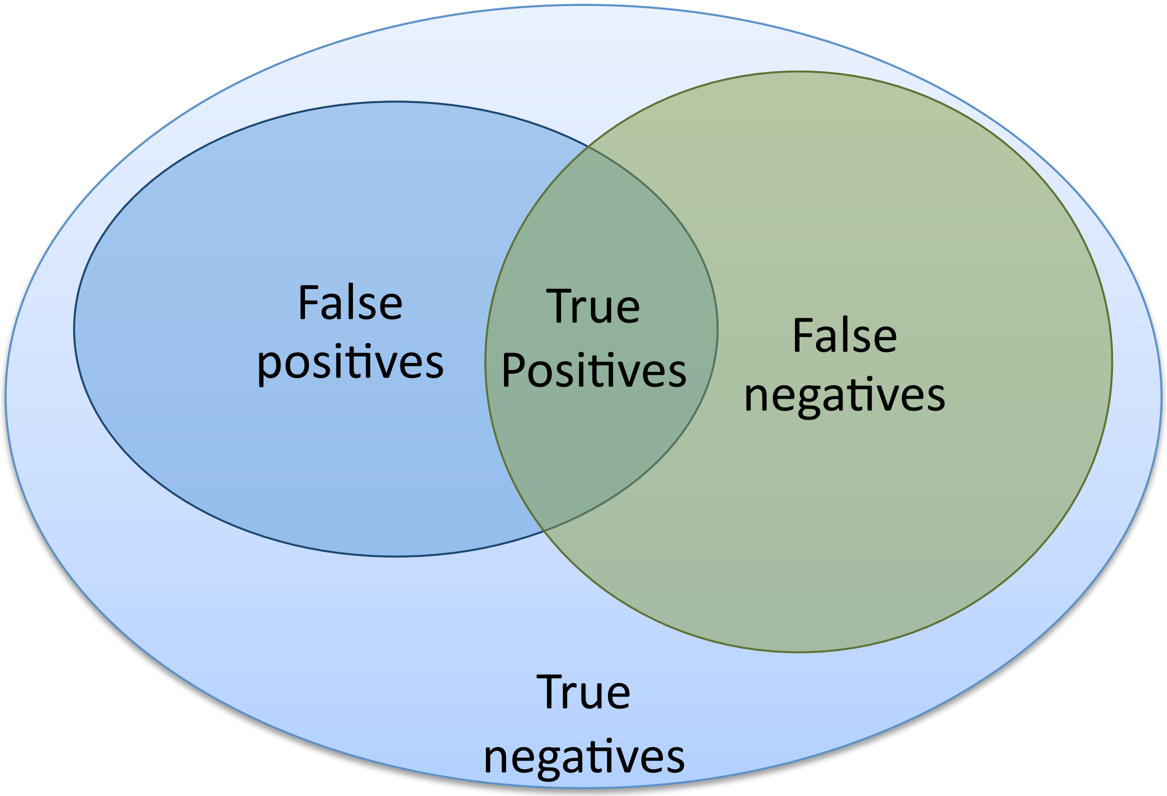
- Equal weights on relevant and irrelevant documents
- #relevant documents usually very small compared to total #documents → insensitive to number of correct decisions



100 relevant documents

Algorithm just returning 0 documents has accuracy of 99900/100000 = 0.99 and error = 0.01

Precision and Recall



$$P = \frac{TP}{TP + FP}$$

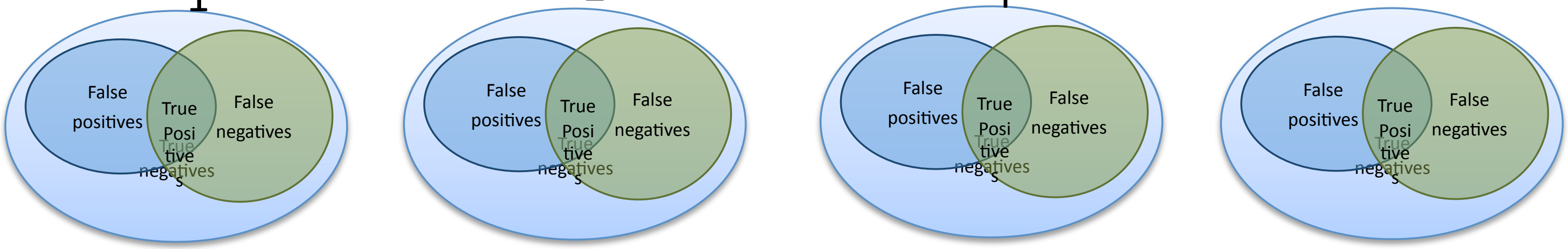
$$R = \frac{TP}{TP + FN}$$

- Precision P: measure for exactness
- Recall R: measure for completeness

Microaveraging

- Sum over all individual decisions

C_1 C_2 C_i C_n

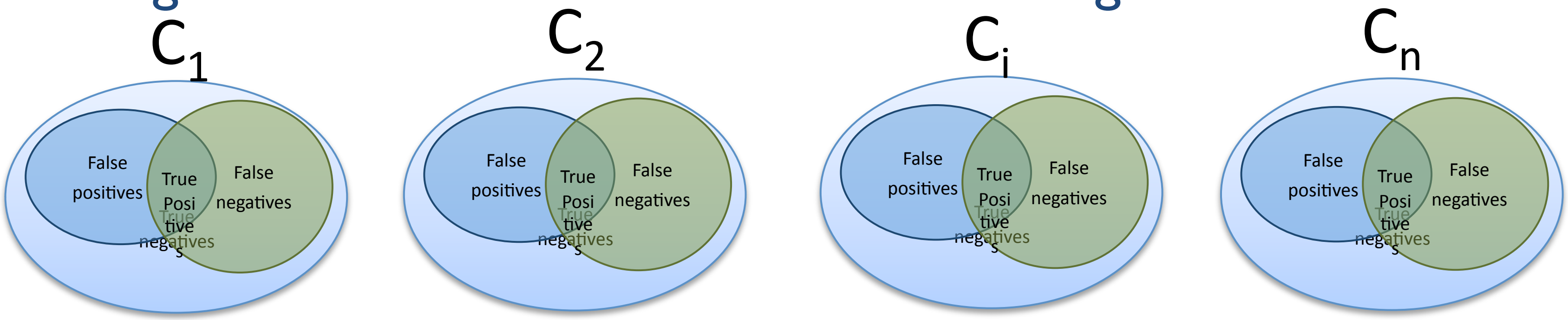


$\rightarrow TP_1, NP_{1,}$ FT_1, FN_1 $\rightarrow TP_2, NP_{2,}$ FT_2, FN_2 $\rightarrow TP_i, NP_{i,}$ FT_i, FN_i $\rightarrow TP_n, NP_{n,}$ FT_n, FN_n

- $\text{Precision}^{\mu} = \Sigma(TP_i) \,/\, \Sigma(TP_i+FP_i)$
- $\text{Recall}^{\mu} = \Sigma(TP_i) \,/\, \Sigma(TP_i+FN_i)$

Macroaveraging

- Average over the results of the different categories



$\rightarrow P_1$ & R_1 $\rightarrow P_2$ & R_2 $\rightarrow P_i$ & R_i $\rightarrow P_n$ & R_n

- $\text{Precision}^M = (P_1 + P_2 + \ldots + P_n) / |C|$
- $\text{Recall}^M = (R_1 + R_2 + \ldots + R_n) / |C|$

Micro- and Macroaveraging

- Difference can be large
- Macroaveraging gives equal weights to each class
- Microaveraging gives equal weights to each classification decision → higher weight on larger classes

Conclusion

- Applications for Text Classification in the Project group
  - Preprocessing task (filter documents, find documents suitable for further Information Extraction tasks)
  - Classifiy documents by
    - Document type → requirements, documentation?
    - Suitable model structure (behavioral, structural) → UML?
    - Combined category search

Questions?