INF4080 – 2019 FALL
NATURAL LANGUAGE PROCESSING

Jan Tore Lønning

(Mostly Text) Classification, Naive Bayes

Lecture 4, 5 Sep

Today - Classification

- Motivation
- Classification of classification
- Some simple examples
- Set-up of experiments
- Evaluation
- Naive Bayes classifiers (two types)

Motivation

Classification

- Jurafsky and Martin, 3.ed. Ch. 4
  Naive Bayes Classification and Sentiment
  - slides 1-7
- NLTK book, Ch. 6

Classification

Classification

- Can be rule-based, but mostly machine learned
- Text classification is a sub-class

- Text classification examples:
  - Spam detection
  - Genre classification
  - Language classification
  - Sentiment analysis:
    - Positive-negative

- Other types of classification:
  - Word sense disambiguation
  - Sentence splitting
  - Tagging
  - Named-entity recognition

Machine learning

1. Supervised
    1. Classification (categorical)
    2. Regression (numerical)
2. Unsupervised
3. Semi-supervised
4. Reinforcement learning

- Supervised:
    - Given classes
    - Given examples of correct classes
- Unsupervised:
    - Construct classes

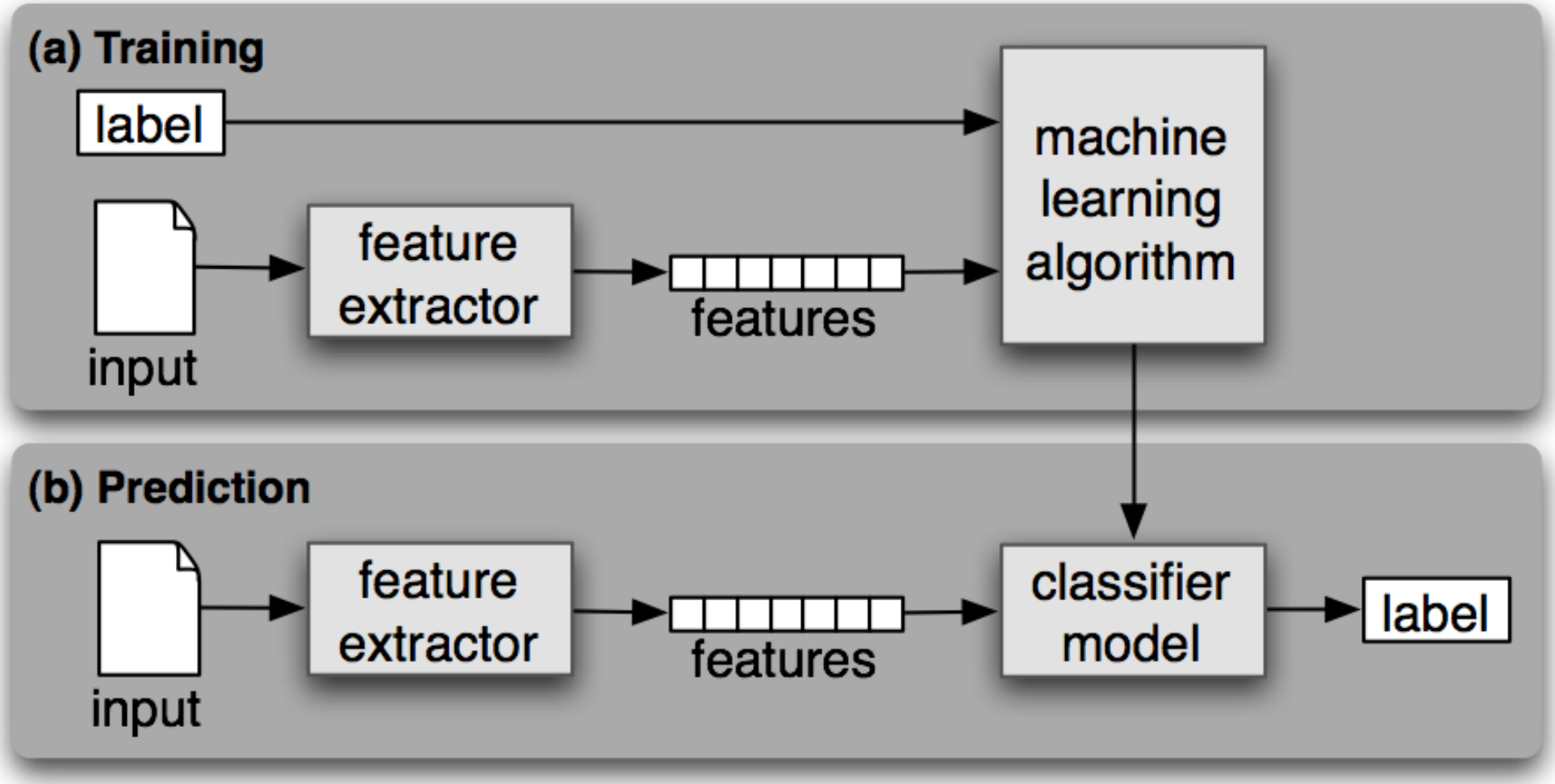
A variety of ML classifiers

- k-Nearest Neighbors
- Rocchio
- Naive Bayes
- Logistic regression (Maximum entropy)
- Support Vector Machines
- Decision Trees
- Perceptron
- Multi-layered neural nets ("Deep learning")

We will (at least) meet

- k-Nearest Neighbors
- Rocchio
- Naive Bayes
- Logistic regression (Maximum entropy)
- Support Vector Machines
- Decision Trees
- Perceptron
- Multi-layered neural nets ("Deep learning")

Classification



Supervised classification

- Given
  - a well-defined set of observations, O
  - a given set of classes, $S=\{s_1, s_2, \ldots, s_k\}$
- Goal: a classifier, γ, a mapping from O to S
- For supervised training one needs a set of pairs from OxS

Task	O	S
Spam classification	E-mails	Spam, no-spam
Language clssification	Pieces of text	Arabian, Chinese, English, Norwegian, …
Word sense disambi-guation	Occurrences of ”bass”	Sense1, …, sense8

Features

- To represent the objects in O, extract a set of features
- Be explicit:
  - Which features
  - For each feature
    - The type
      - Categorical
      - Numeric (Discrete/Continuous)
    - The value space

O: person
Features:
- height
- weight
- hair color
- eye color
- …

O: email
Features:
- length
- sender
- contained words
- language
- …

Cf. First lecture
Classes and features are both attributes of the observations

Supervised classification

- A given set of classes, $S=\{s_1, s_2, \ldots, s_k\}$
- A well defined class of observations, O

- Some features $f_1, f_2, \ldots, f_n$
- For each feature: a set of possible values $V_1, V_2, \ldots, V_n$
- The set of feature vectors: $V= V_1 \times V_2 \times \ldots \times V_n$
- Each observation in O is represented by some member of V:
  - Written $(v_1, v_2, \ldots, v_n)$, or
  - $(f_1=v_1, f_2=v_2, \ldots, f_n=v_n)$
- A classifier, γ, can be considered a mapping from V to S

Examples

Language classifier

- C = {English, Norwegian,…}
- O is the set of strings of letters
- f_1 is last letter of o
- V_1= {a, b, c,…, å}
- f_2 is the last two letters
- V_2 is all two letter combinations
- f_3 is the length of o,
- V_3 is 1, 2, 3, 4, …

Word sense disambiguation

- C = {fish, music}
- O: all occurrences of ”bass”
- A finite set of indicator words {w_1 = *fishing*, w_2 = *big*, …, w_{11} = *guitar*, w_{12} = *band*}
- f_i= 1 iff word w_i occurs in same sentence as ”bass”, 0, otherwise
- V_1=V_2=…=V_{12}={1,0}
- Example:
  - o = (0,0,0,1,0,0,0,0,0,0,1,0)
  - o = ($f_{fishing}$=0, …, f_{guitar}=1, f_{band}=0)

Simple examples from NLTK

NLTK-example 1: names

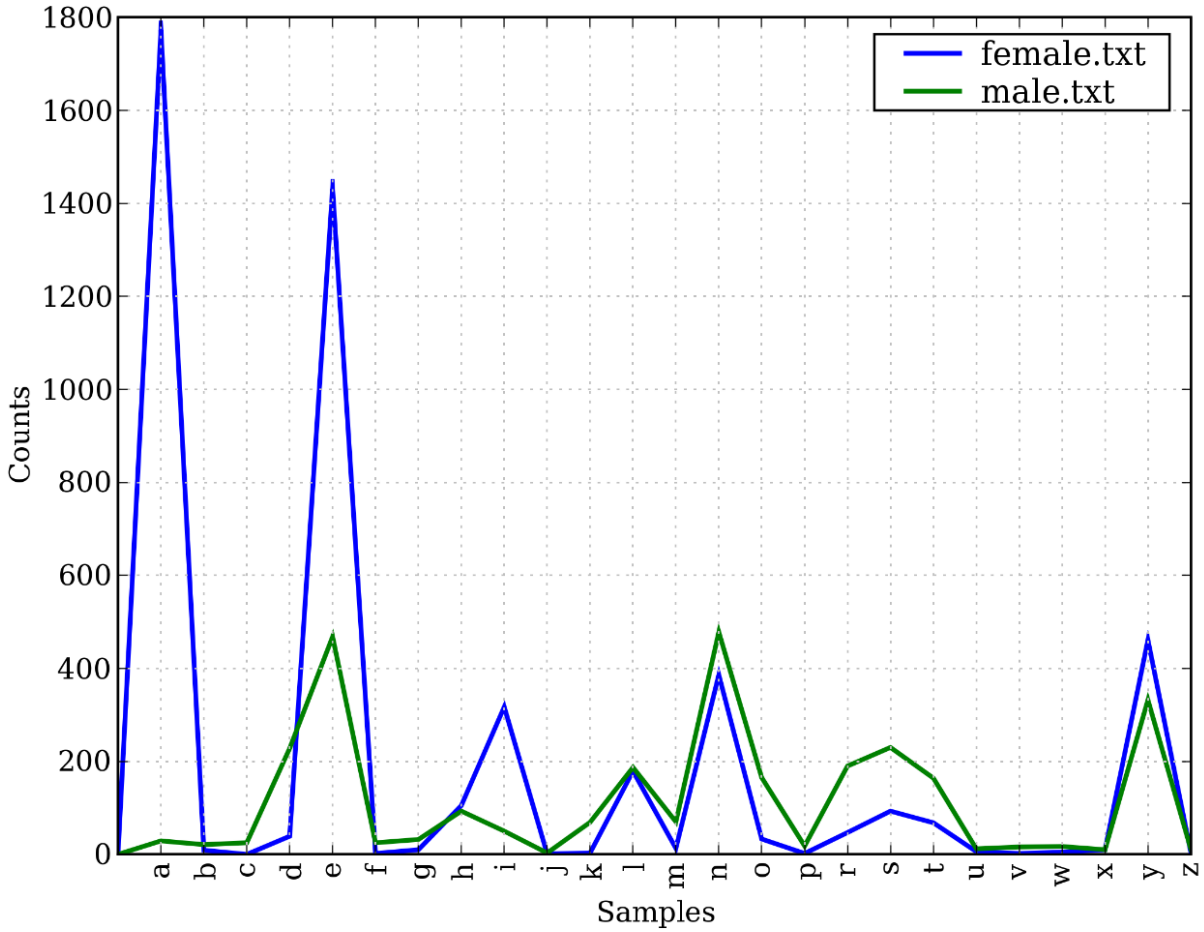

```
In [2]: def gender_features(word):
   ...:     return {'last letter': word[-1]}

In [3]: gender_features('Shrek')
Out[3]: {'last letter': 'k'}

In [4]: from nltk.corpus import names

In [5]: labeled_names =
  ([(name, 'male') for name in names.words('male.txt')] +
   [(name, 'female') for name in names.words('female.txt')])
```

NLTK: names


female.txt
male.txt
Counts
Samples
0
200
400
600
800
1000
1200
1400
1600
1800
a b c d e f g h i j k l m n o p r s t u v w x y z

NLTK-example 1, contd.


```
In [6]: import random
In [8]: random.shuffle(labeled_names)
In [9]: featuresets =
            [(gender_features(n), gender)
            for (n, gender) in labeled_names]
In [10]: train_set, test_set =
           featuresets[500:], featuresets[:500]
```

When you conduct several experiments, use the same split so you can compare the results.

Split before you extract features

NLTK-example1, contd.


```
In [11]: classifier = nltk.NaiveBayesClassifier.train(train_set)
In [12]: classifier.classify(gender_features('Neo'))
Out[12]: 'male'
In [13]: classifier.classify(gender_features('Ada'))
Out[13]: 'female'
In [31]: print(nltk.classify.accuracy(classifier, test_set))
0.79
```

Why do I get 0.79 and the book 0.75?

Example 1 ctd.

- A given set of classes, $S=\{s_1, s_2, \ldots, s_k\}$ = {'male', 'female'}
- A well defined class of objects, O = {'Ada', 'Albert', ...} = all strings of letters

- Some features $f_1, f_2, \ldots, f_n$, only f_1 'last_letter'
- For each feature: a set of possible values $V_1, V_2, \ldots, V_n$, V_1 ={a, b, c, …., z}
- The set of feature vectors: $V= V_1 \times V_2 \times \ldots \times V_n$
- Each object in O is represented by some member of V:
  - Written $(v_1, v_2, \ldots, v_n)$, or (e.g. 'u')
  - $(f_1=v_1, f_2=v_2, \ldots, f_n=v_n)$ (e.g. last_letter: 'u')
- A classifier, γ, can be considered a mapping from V to S

NLTK-example 2


```
In [56]: def gender_features2(name):
    ...:     features = {}
    ...:     features["first_letter"] = name[0].lower()
    ...:     features["last_letter"] = name[-1].lower()
    ...:     for letter in 'abcdefghijklmnopqrstuvwxyz':
    ...:         features["count({})".format(letter)] = name.lower().count(letter)
    ...:         features["has({})".format(letter)] = (letter in name.lower())
    ...:     return features
```

What are the features here?

- How many?
- What are their resp. value spaces?

Comparing features

- NLTK-book printed:
  - gender_features (gf1) yields acc 0.758
  - gender_features2 (gf2) yields acc 0.748
- Indicates
  - More features aren't always better
  - Danger that gender_features2 "is overfitting":
    - Adapt itself too much to the training set
- Web edition: gf1_acc: 0.77, gf2_acc: 0.768
- We: gf1_acc: 0.79, gf2_acc: 0.78

A more complex picture

- 10 experiments
- Do not draw hasty conclusions from small differences
- Variation
- We will later consider how statistics may tell us which differences are significant

Accuracy:

Exp.no	gf1	gf2
1	0.760	0.756
2	0.770	0.784
3	0.782	0.774
4	0.772	0.796
5	0.744	0.744
6	0.760	0.792
7	0.776	0.754
8	0.782	0.784
9	0.774	0.774
10	0.772	0.794

Movie reviews


```
from nltk.corpus import movie_reviews

documents = [(list(movie_reviews.words(fileid)), category)
             for category in movie_reviews.categories()
             for fileid in movie_reviews.fileids(category)]

random.shuffle(documents)

all_words = nltk.FreqDist(w.lower() for w in movie_reviews.words())

word_features = [w for (w,_) in all_words.most_common(2000)]

wrong_features = list(all_words)[:2000] #Wrong (earlier version)
```

Movie reviews ctd.


```
def document_features(word_features, document):
      document_words = set(document)
      features = {}
      for word in word_features:
           features['contains({})'.format(word)] =  (word in document_words)
                                                     #True or False
      return features

featuresets = [(document_features(word_features, d), c)  for (d,c) in documents]

train_set, test_set = featuresets[100:], featuresets[:100]
classifier = nltk.NaiveBayesClassifier.train(train_set)

print(nltk.classify.accuracy(classifier, test_set))
 0.83
```

Movie reviews ctd.

Properties

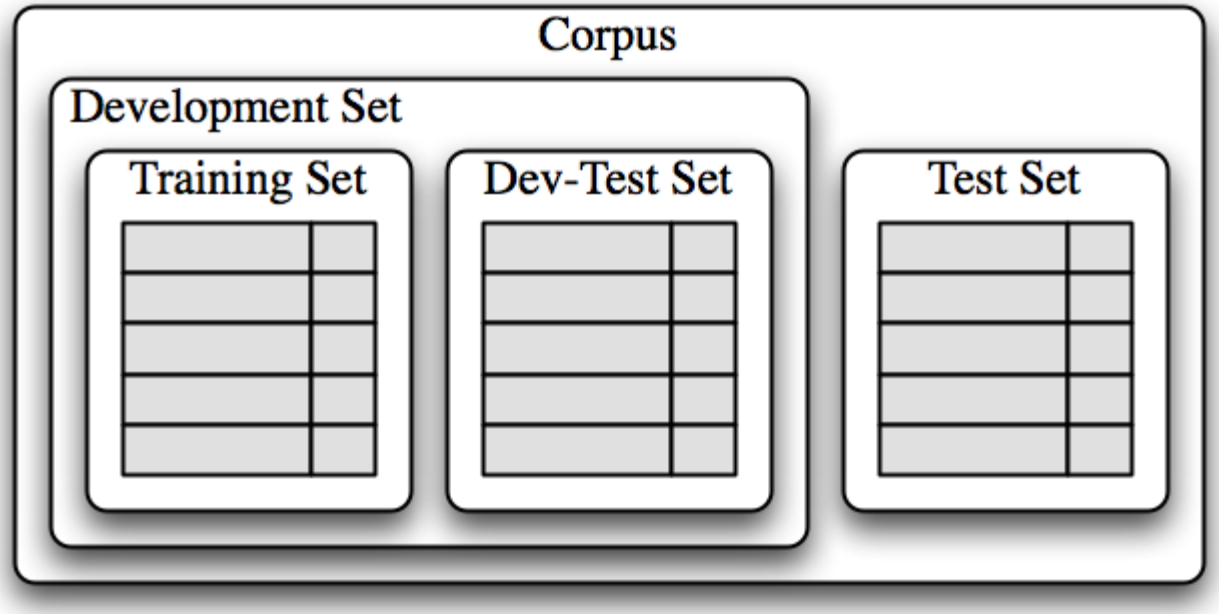
- Two classes: ‘neg’, ‘pos’
- Features’:
  - 2000 most frequent words in corpus
- Values: True/False
  - Don't count number of occurrences in each document
  - All features (words) not in document gets value ”False” for the document

Comments

- Strictly speaking, the "most frequent" should be counted from training data only

Set-up for experiments

Set-up for experiments



- Before you start: split into development set and test set.
- Hide the test set
- Split development set into Training and Development-Test set
- Use training set for training a learner
- Use Dev(-Test) for repeated evaluation in the test phase
- Finally test on the test set!

Procedure

1. Train classifier on training set
2. Test it on dev-test set
3. Compare to earlier runs, is this better?
4. Error analysis: What are the mistakes (on dev-test set)
5. Make changes to the classifier
6. Repeat from 1

==========

- When you have run empty on ideas, test on test set. Stop!

Cross-validation

- Small test sets ➔ Large variation in results
- N-fold cross-validation:
  - Split the development set into n equally sized bins
    - (e.g. n = 10)
  - Conduct n many experiments:
    - In experiment m, use part m as test set and the n-1 other parts as training set.
  - This yields n many results:
    - We can consider the mean of the results
    - We can consider the variation between the results.
      - Statistics!

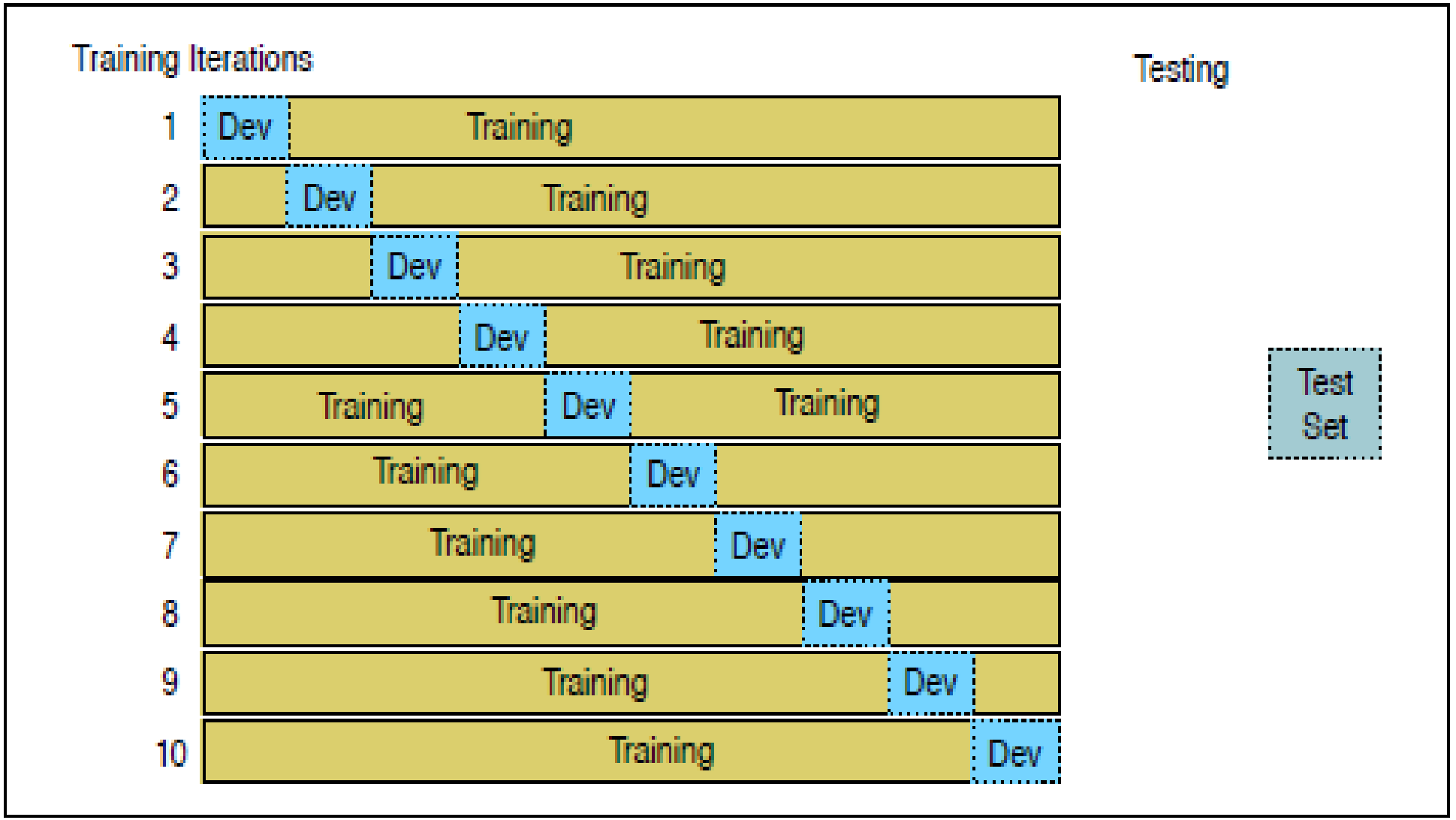


Figure 6.7 10-fold crossvalidation

Evaluation

Evaluation measure: Accuracy

- What does accuracy 0.81 tell us?
- Given a test set of 500 sentences:
  - The classifier will classify 405 correctly
  - And 95 incorrectly
- A good measure given:
  - The 2 classes are equally important
  - The 2 classes are roughly equally sized
  - Example:
    - Woman/man
    - Movie reviews: pos/neg

But

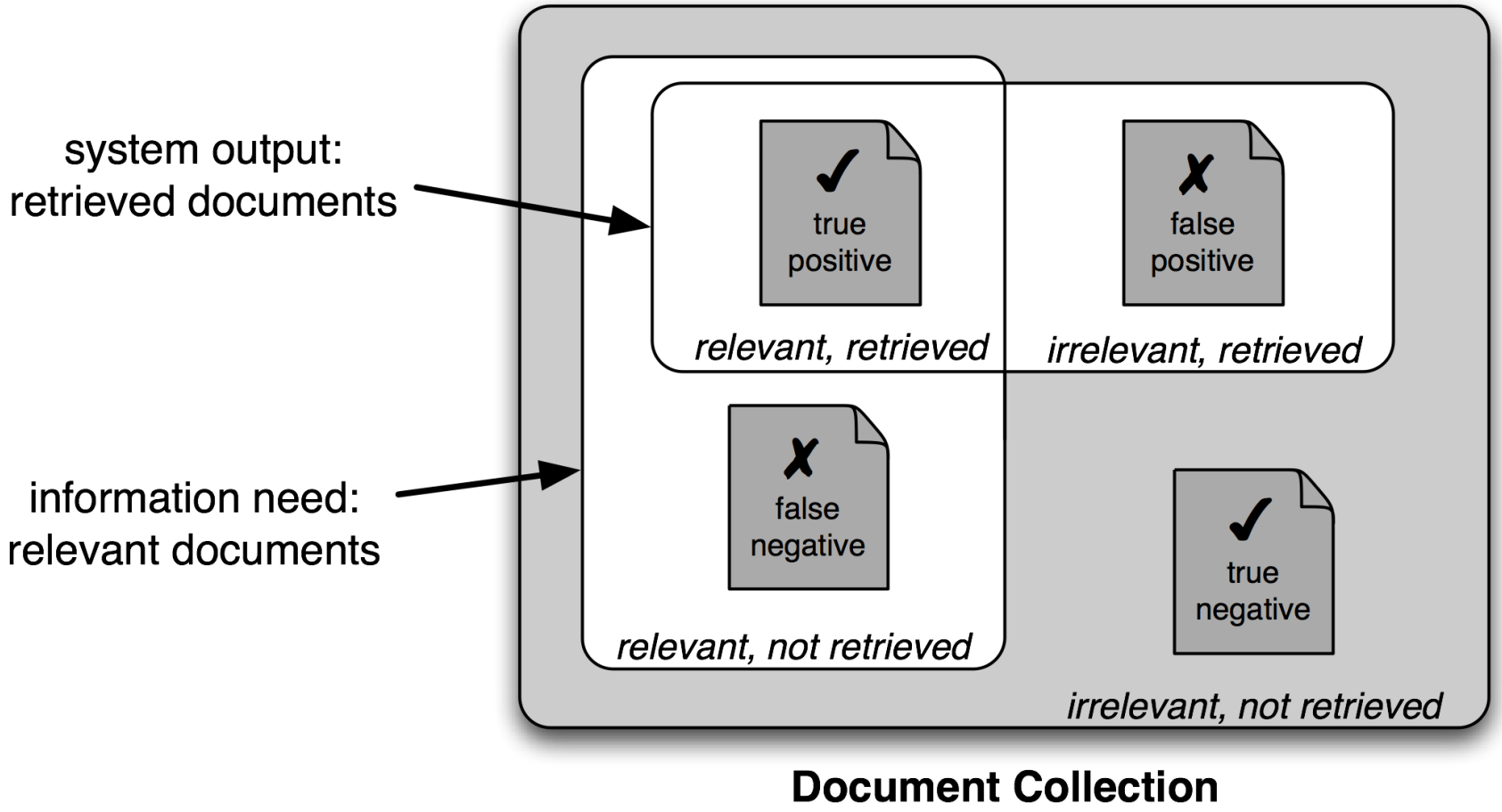
- For some tasks the classes aren't equally important
  - Worse to loose an important mail than to receive yet another spam mail
- For some tasks the different classes have different sizes.

Information retrieval (IR)

- Traditional IR, e.g. a library
  - Goal: Find all the documents on a particular topic out of 100 000 documents,
    - Say there are 5
  - The system delivers 10 documents: all irrelevant
    - What is the accuracy?

- For these tasks, focus on
  - The relevant documents
  - The documents returned by the system
- Forget the
  - Irrelevant documents which are not returned

IR - evaluation



Confusion matrix

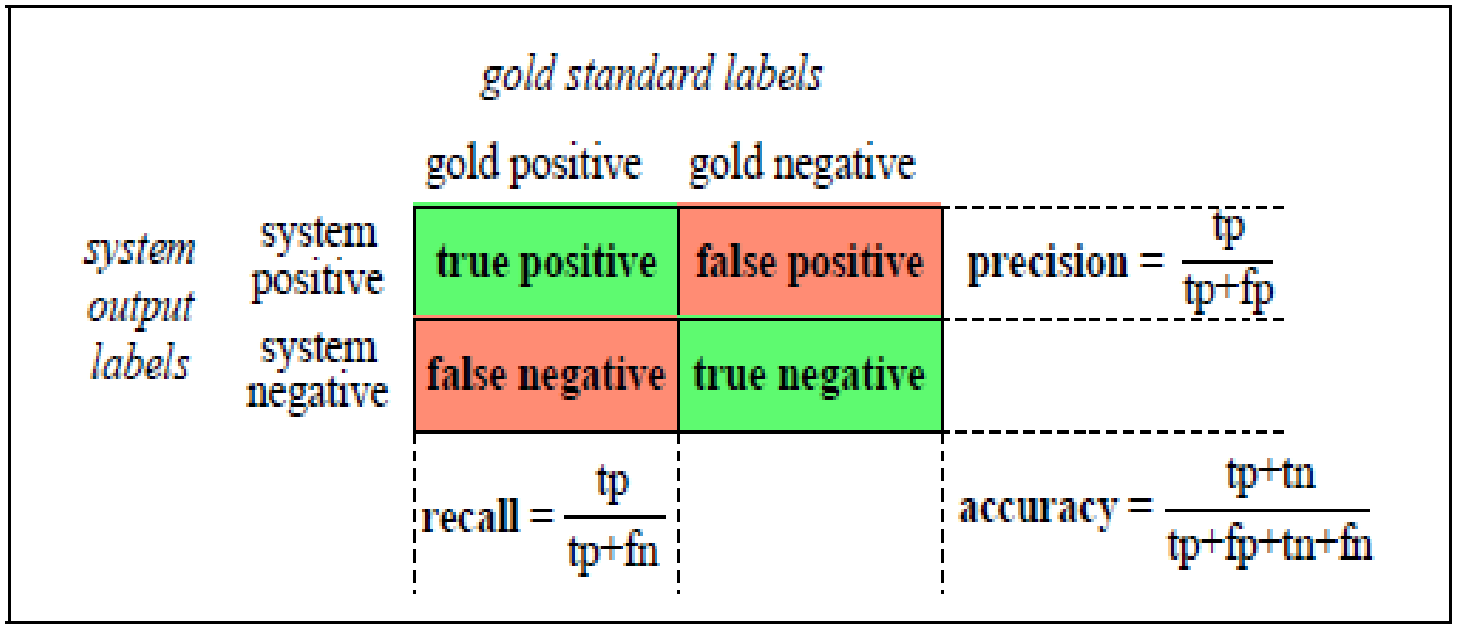


Figure 6.4 Contingency table

- Beware what the rows and columns are:
  - NLTKs ConfusionMatrix swaps them compared to this table

Evaluation measures

		Is in C	
		Yes	NO
Classifier	Yes	tp	fp
	No	fn	tn

- Accuracy: (tp+tn)/N
- Precision:tp/(tp+fp)
- Recall: tp/(tp+fn)

- F-score combines P and R
- $F_1 = \frac{2PR}{P+R} \left(= \frac{1}{\frac{\frac{1}{R}+\frac{1}{P}}{2}} \right)$
- F_1 called "harmonic mean"
- General form
  - $F = \frac{1}{\alpha\frac{1}{P}+(1-\alpha)\frac{1}{R}}$
  - for some $0 < \alpha < 1$

Confusion matrix

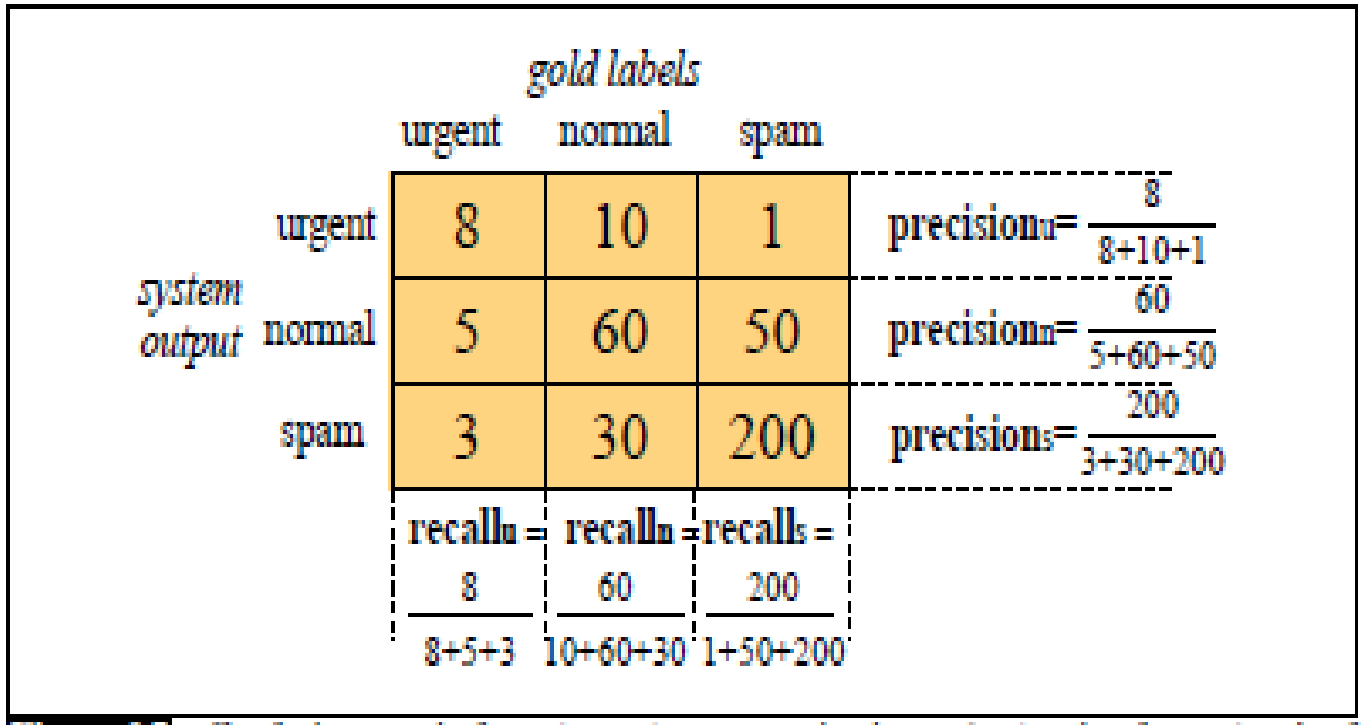


Figure 6.5 Confusion matrix for a three-class categorization task, showing for each pair of classes (c_1, c_2), how many documents from c_1 were (in)correctly assigned to c_2

- Precision, recall and f-score can be calculated for each class against the rest

Naive Bayes

Naive Bayes: Decision

- Given an observation
  - $\langle f_1 = v_1, f_2 = v_2, \ldots, f_n = v_n \rangle$
- Consider
  - $P(s_m \mid \langle f_1 = v_1, f_2 = v_2, \ldots, f_n = v_n \rangle)$ for each class s_m
- Choose the class with the largest value, in symbols

$$\arg\max_{s_m \in S} P(s_m \mid \langle f_1 = v_1, f_2 = v_2, \ldots, f_n = v_n \rangle)$$

- i.e. choose the class for which the observation is most likely

Naive Bayes: Model

- Bayes formula
  - $P(s_m \mid \langle f_1 = v_1, f_2 = v_2, \ldots, f_n = v_n \rangle) = \frac{P(\langle f_1 = v_1, f_2 = v_2, \ldots, f_n = v_n \rangle \mid s_m) P(s_m)}{P(\langle f_1 = v_1, f_2 = v_2, \ldots, f_n = v_n \rangle)}$
- Sparse data, we may not even have seen
  - $\langle f_1 = v_1, f_2 = v_2, \ldots, f_n = v_n \rangle$
- We assume (wrongly) independence
  - $P(\langle f_1 = v_1, f_2 = v_2, \ldots, f_n = v_n \rangle \mid s_m) \approx \prod_{i=1}^{n} P(f_i = v_i \mid s_m)$
- Putting together, choose
  - $\arg\max_{s_m \in S} P(s_m \mid \langle f_1 = v_1, f_2 = v_2, \ldots, f_n = v_n \rangle) \approx \arg\max_{s_m \in S} P(s_m) \prod_{i=1}^{n} P(f_i = v_i \mid s_m)$

Naive Bayes: Calculation

- $\arg\max_{s_m \in S} P(s_m \mid \langle f_1 = v_1, f_2 = v_2, \ldots, f_n = v_n \rangle) \approx \arg\max_{s_m \in S} P(s_m) \prod_{i=1}^{n} P(f_i = v_i \mid s_m)$

- For calculations
  - avoid underflow, use logarithms
  - $\arg\max_{s_m \in S} P(s_m) \prod_{i=1}^{n} P(f_i = v_i \mid s_m) =$

    $\arg\max_{s_m \in S} \left(\log \left(P(s_m) \prod_{i=1}^{n} P(f_i = v_i \mid s_m) \right) \right)$

    $= \arg\max_{s_m \in S} \left(\log(P(s_m)) + \sum_{i=1}^{n} \log(P(f_i = v_i \mid s_m)) \right)$

Naive Bayes, Training 1

- Maximum Likelihood
- $$\hat{P}(s_m) = \frac{C(s_m, o)}{C(o)}$$
  - where C(s_m, o) are the number of occurrences of observations o in class s_m
- Observe what we are doing:
  - We are looking for the true probability $P(s_m)$
  - $\hat{P}(s_m)$ is an approximation to this, our best guess from a set of observations
  - Maximum likelihood means that it is the model which makes the set of observations we have seen, most likely

Naive Bayes: Training 2

- Maximum Likelihood
- $$\hat{P}(f_i = v_i \mid s_m) = \frac{C(f_i = v_i, s_m)}{C(s_m)}$$
  - where $C(f_i=v_i, s_m)$ is the number of occurrences of observations o
    - where the observation o belongs to class s_m
    - and the feature f_i takes the value v_i
  - $C(s_m)$ is the number of occurrences belonging to class s_m

Example: movie reviews NLTK (Bernoulli)

- '*pitt*' occurs in 15 'pos' and 6 'neg' reviews
- $\hat{P}(pitt = True|pos) = \frac{15}{959}$ $\hat{P}(pitt = True|neg) = \frac{6}{941}$
- $\hat{P}(pitt = False|pos) = \frac{944}{959}$ $\hat{P}(pitt = False|neg) = \frac{935}{941}$

Example: more features

- In [63]: pos_docs['pitt']
- Out[63]: 15

- In [64]: neg_docs['pitt']
- Out[64]: 6

- In [65]: neg_docs['spacey']
- Out[65]: 4

- In [66]: pos_docs['spacey']
- Out[66]: 17

- In [71]: pos_docs['terrible']
- Out[71]: 26

- In [72]: neg_docs['terrible']
- Out[72]: 85

- In [73]: neg_docs['terrific']
- Out[73]: 19

- In [74]: pos_docs['terrific']
- Out[74]: 75

$pitt = True, terrible = True, terrific = False$

'pos'

- $\hat{P}(pos) = \frac{959}{1900}$
- $\hat{P}(pitt = True|pos) = \frac{15}{959}$
- $\hat{P}(terrible = True|pos) = \frac{26}{959}$
- $\hat{P}(terrifi = False|pos) = \frac{959-75}{959}$
- $\hat{P}(pos \mid pitt = True, terrible = True, terrific = False) = k\frac{959}{1900} \times \frac{15}{959} \times \frac{26}{959} \times \frac{884}{959} = k0.00020$

'neg'

$\hat{P}(neg) = \frac{941}{1900}$

$\hat{P}(pitt = True|neg) = \frac{6}{941}$

$\hat{P}(terrible = True|neg) = \frac{85}{941}$

$\hat{P}(terrifi = False|pos) = \frac{941-19}{941}$

$\hat{P}(neg \mid pitt = True, terrible = True, terrific = False) = k\frac{941}{1900} \times \frac{6}{941} \times \frac{85}{941} \times \frac{922}{941} = k0.00028$

($k = 1/P(pitt = True, terrible = True, terrific = False)$, the same for both classes)

The two models

- Bernoulli
  - NLTK book, Sec. 6.1, 6.2, 6.5
  - Jurafsky and Martin, 2.ed, sec. 20.2, WSD
- Multinomial model
  - For text classification
  - Related to n-gram models (next week)
  - Jurafsky and Martin, 3.ed, sec. 4, Sentiment analysis
- Both
  - Manning, Raghavan, Schütze, *Introduction to Information Retrieval*, Sec. 13.0-13.3

Comparison

Bernoulli

- Registers whether a term is present or not
- Considers both
  - The present terms
  - The absent terms
- Suitable for various tasks

Multinomial

- Counts how many times a term is present
- Considers
  - only the present terms
  - Ignores absent terms
- Tailor-made for text classification

Multinomial text classification

- Build a language model for each class
- Score the document according to the different classes
- Choose the class with the best score

Multinomial NB: Decision

$$\arg\max_{s_m \in S} P\left(s_m \mid \langle f_1 = v_1, f_2 = v_2, \ldots, f_n = v_n \rangle\right) \approx \arg\max_{s_m \in S} P(s_m) \prod_{i=1}^{n} P\left(f_i = v_i \mid s_m\right)$$

- f_i refers to position i in the text, v_i is the word occurring in this position
- Simplifying assumption: a word is equally likely in all positions
- Hence we count how many times each word occurs in the text

$$\arg\max_{s_m \in S} P(s_m) \prod_{i=1}^{n} P\left(f_i = v_i \mid s_m\right) = \arg\max_{s_m \in S} P(s_m) \prod_{i=1}^{n} P\left(v_i \mid s_m\right)$$

Multinomial NB: Training

- $\hat{P}(s_m) = \frac{C(s_m, o)}{C(o)}$
  - where $C(s_m, o)$ is the number of occurrences of objects o in class s_m
- $\hat{P}(w_i \mid s_m) = \frac{C(w_i, s_m)}{\sum_j C(w_j, s_m)}$
  - where $C(w_i, s_m)$ is the number of occurrences of word w_i in all texts in class s_m
  - $\sum_j C(w_j, s_m)$ is the total number of words in all texts in class s_m
- Bernoulli counts the number of objects/texts where w_i occurs
- Multinomial counts the number of occurrences of w_i.

Example: movie reviews, multinomial

- '*pitt*' occurs in 15 'pos' and 6 'neg' reviews
- 'pitt' occurs 31 times in the 'pos' reviews and 25 times in the negative reviews
- There are 798,742 words in the 'pos' reviews and 705,726 in the 'neg' reviews
- $\hat{P}(w = pitt|pos) = \frac{31}{798\,742}$ $\hat{P}(w = pitt|neg) = \frac{25}{705\,726}$

$3 \times pitt, \quad 2 \times terrible, \quad 0 \times terrific$

'pos'

- $\hat{P}(pos) = \frac{959}{1900}$
- $\hat{P}(w = pitt|pos) = \frac{31}{798\,742}$
- $\hat{P}(w = terribl|pos) = \frac{26}{798\,742}$
- $\hat{P}(pos \mid 3 \times pitt, 2 \times terrible, 0 \times terrific) =$

$$k' \frac{959}{1900} \times \left(\frac{31}{798\,742}\right)^3 \times \left(\frac{26}{798\,742}\right)^2$$
$$= k'\, 3.12 \times 10^{-23}$$

'neg'

$$\hat{P}(neg) = \frac{941}{1900}$$

$$\hat{P}(w = pitt|neg) = \frac{25}{705\,726}$$

$$\hat{P}(w = terribl|neg) = \frac{104}{705726}$$

$$\hat{P}(pos \mid 3 \times pitt, 2 \times terrible, 0 \times terrific) =$$

$$k' \frac{941}{1900} \times \left(\frac{25}{705\,726}\right)^3 \times \left(\frac{104}{705\,726}\right)^2$$
$$= k'\, 4.78 \times 10^{-22}$$