Chapter 10: Text Categorization

Overview

- Definitions
- Oldest approaches: symbolic techniques
- Feature selection and extraction
- Machine learning techniques:
    - learning of rules and trees
    - naive Bayes classification
    - support vector machine
- Text categorization results
- Hierarchical categorization

Definitions

- Text categorization = assignment of controlled language descriptors to texts
  - usually categorization of complete document texts
  - descriptors or labels
    - concept terms, subject or classification codes: $C = \{C_1, C_2, \ldots, C_m\}$
    - labels might be organized in hierarchy
  - when classes are not mutually exclusive:
    - often seen as a two-class learning problem
    - or as a multi-label multi-class learning problem (not treated in this chapter)

Text categorization examples

- Descriptors are most often topics (hierarchically ordered) such as Yahoo-categories

  e.g., finance, sports, news>world>asia>business
- Descriptors may represent genres

  e.g., editorial, review, news
- Descriptors may reflect opinions

  e.g., positive review, neutral review, negative review
- Descriptors may be binary

  e.g., relevant or *not relevant*

  e.g., spam or *not spam*

  e.g., contains adult language or *doesn't*

Oldest approaches: symbolic techniques

- Oldest approach (1980s-beginning of 1990s) uses knowledge base of classification patterns:
  - patterns:
    - words and phrases
    - possibly constrained by logical operators (e.g., ¬, ∧, ∨)
    - possibly constrained by frequency of occurrence
  - represented with decision rules or trees, or frames

Oldest approaches: symbolic techniques

- **Decision or production rule**:

  IF <condition is true>

  THEN <assign category>

  e.g., in common logical formalism (propositional or first order predicate logic)

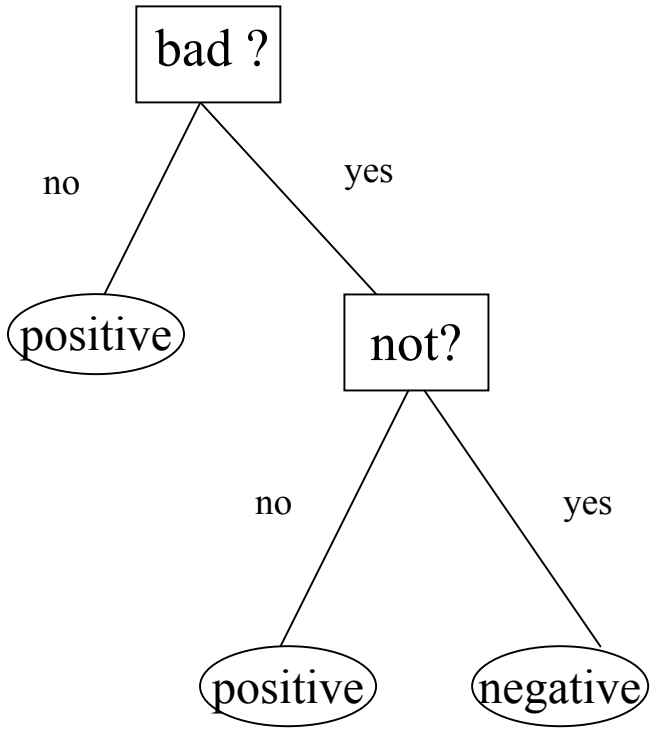
  e.g.,

  IF « deficit » ∧ « import » ∧ « export »

  THEN CLASS = trade

Oldest approaches: symbolic techniques

- **Decision tree:**
  - Consists of nodes and branches
  - Each node, except for terminal nodes or leaves, represents a test or decision and branches into subtrees for each possible outcome of the test
  - Each leaf is associated with a particular class



Oldest approaches: symbolic techniques

- **Frames**:
  - feature slots: fixed for a particular frame
  - feature values: fill the slots, certain constraints can be placed
  - can be connected in a semantic net
  - advantages:
    - default values, inherited values

Define type for *book_review*
begin
subtype of *Article*
features:
author: John Smith;
cue_pattern: *isbn_template*;
...
end;

Symbolic techniques

- Successful technique*:*
  - categorizing office documents, news stories, messages,...
  - recall and precision rates (compared to human assignment) in the 90%: approximate the quality of human assignment of descriptors

    => surface text features can be identified that successfully discriminate the subject and classification codes linked to a text
- But:
  - primarily semantic knowledge of a particular domain of discourse
  - in heterogeneous domains: (semi)-supervised **machine learning** methods provide a very valuable alternative

Feature selection

1) Unsupervised selection:

- **words** (e.g., unigrams): elimination of stopwords, terms with low weights (weights based on frequency, time-sensitiveness, …)
- selection of words based on a domain-specific dictionary, based on location in discourse, etc.
- association patterns:
  - **frequent item set patterns**
  - **hyperclique patterns**

- **Frequent (item set) patterns**:
  - item set *a* (here feature set) that satisfies minimum relative support, i.e., $\frac{|D_a|}{|D|} \geq \theta_0, \quad 0 \leq \theta_0 \leq 1$ where |*D*|= number of documents in the collection and |D_a|= number of documents in the collection containing frequent pattern *a*
  - efficient computations of frequent patterns using the *a priori* algorithm and variants
- **Hyperclique patterns**:
  - features in the item set are highly correlated with each other (e.g., collocations of words)
  - efficient algorithms for their identification

Feature selection

2) Supervised selection:
   - compute the **relevance score** of a text feature and select features with high relevance score
   - many algorithms that compute correlation between term and class, e.g.,:
     - χ^2 (*chi-square*) measure
     - information gain
     - ...

Feature selection

- Caution in removing features from texts **!!!**:
  - words that seem to have low overall content bearing value can be an important category indicator especially in combination with other terms

χ^2 measure

- χ^2 measures the degree of dependence (lack of independence) between an observed probability distribution and an expected distribution
- In the context of text categorization, it measures the **fit between the observed frequency of a feature in the training example set and its expected frequency**, i.e., the feature occurs with equal frequency in texts that are relevant for a specific class and in texts that are are not relevant for this class

χ^2 measure

- For each feature t (usually word) we compute the χ^2 value for each class C_j using a contingency table:

$$\chi^2(t,C_j) = \frac{n(n_{r+}n_{n-} - n_{r-}n_{n+})^2}{(n_{r+}+n_{r-})(n_{n+}+n_{n-})(n_{r+}+n_{n+})(n_{r-}+n_{n-})}$$

where $n =$ number of object measured (here documents in the training set)

χ^2 measure

	Texts relevant for class C_j	Texts not relevant for class C_j	
With feature t	n_{r+}	n_{n+}	$n_{r+} + n_{n+}$
Without feature t	n_{r-}	n_{n-}	$n_{r-} + n_{n-}$
	$n_{r+} + n_{r-}$	$n_{n+} + n_{n-}$	$n_{r+} + n_{n+} + n_{r-} + n_{n-} = n$

χ^2 measure

- The formula is derived by calculating the difference between the observed frequency and expected frequency in each of the 4 cases
- For example, when there are n_{r+} documents of class C_j with feature *t:*
  - the observed frequency of t is n_{r+}
  - the expected frequency is: $= \frac{(n_{r+} + n_{r-})(n_{r+} + n_{n+})}{n}$
  - the squared difference of the observed and expected frequency in this cell is:

$$= \frac{(n_{r+} - \frac{(n_{r+} + n_{r-})(n_{r+} + n_{n+})}{n})^2}{\frac{(n_{r+} + n_{r-})(n_{r+} + n_{n+})}{n}}$$

χ^2 measure

- χ^2 value= sum of the squared differences for each cell
- In χ^2 table (here 1 degree of freedom) you can detect at a chosen probability level whether the hypothesis of fit can be accepted or rejected
- In text categorization: features with high χ^2 value are considered to be strongly related to the category
- In feature selection features with highest χ^2 value are selected
- Caution when an expected cell frequency < 5 or $n < 50$

χ^2 measure: example

Magazine articles from Knack, Trends, Weekend Knack, Cash, … written in Dutch

- 1745 training documents with 14 classes
- some χ^2 values of terms for the class **car**:

kleppen (valves): 161.37
cilinder (cylinder): 123.92
pk (horse power): 427.22
toeren (gear): 325.18
topsnelheid (top speed): 173.88
verbruik (consumption): 136.39

prijs (price): 31.13
Chrysler: 5.21
stadsverkeer (city traffic) : 12.33
piano: 0.41
Europa (Europe):0.51
bedrijf (company): 0.008

Entropy

- Measures homogeneity of examples
- Given a collection S of training examples, if the classification can take on m different values, then the entropy of S relative to the m classifications is defined as:

$$Entropy(S) \equiv \sum_{j=1}^{m} -p_j \log_2 p_j$$

where p_j is the proportion of S belonging to class j

Information gain

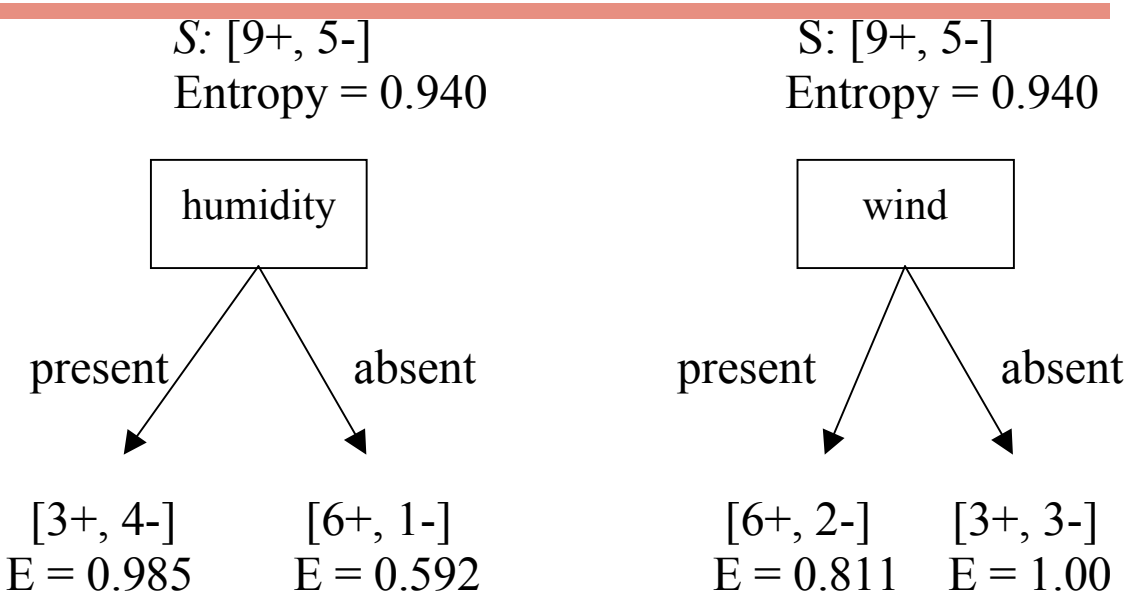
- The information gain of a feature *f* = the **expected reduction in entropy** caused by partitioning the examples according to the value of this feature:

$$Gain(S, f) \equiv Entropy(S) - \sum_{v \in Values(f)} \frac{|S_v|}{|S|} Entropy(S_v)$$

where $Values(f)$ = set of all possible values of feature f

S_v = subset of S for which feature f has value v

Example: which attribute is the best classifier for the class Climate?



Gain (*S*, *humidity*)
= 0.940 –(7/14)0.985– (7/14).0.592
= 0.151

Gain (*S*, *wind*)
= 0.940 – (8/14)0.811– (6/14)1.0
=0.048

Feature extraction

- Transformation of text features: e.g.,
  - stemming of words
  - phrase formation
  - term weighting (e.g., term frequency (*tf*), inverse document frequency (*idf*), and *tf* x *idf*
  - use of thesauri:
    - transforms the original word and phrases with more uniform and more general concepts
    - expansion: e.g., ‘January’, ‘February’, ….,’December’
  - latent semantic indexing, topic models

Machine learning techniques

- Challenges to overcome:
  - high-dimensional feature space (e.g., many words) sparse document vectors
  - very small overlap of document vectors (except for stopwords)
  - often few positive training examples: cues for category often occur only in one or none of the (training) examples
  - most words do occur in medium range of frequency (cf. law of Zipf): problem how to discriminate good and bad category cue terms
  - space and time restrictions

Learning of rules and trees

- = inducing classifying expressions from classified example texts in the form of **decision rules** or **trees**
- Principle:
  - the training examples are represented:
    - as a set of features
    - as a set of relations between features
  - learning = **searching the space** of possible **hypotheses** (decision rules / trees) built with the features in order to obtain a valid hypothesis which is:
    - *complete* when it covers all positive examples
    - *consistent* when it does not cover any negative examples

Learning of rules and trees

- Rules:
  - **General-to-specific search** (1):
    - starts from the most general rule possible (often an empty clause), which is **specialized** at the encounter of a negative example that is covered
    - principle: adding features to the rule
  - **Specific-to-general search** (2)
    - starts with a positive example, which forms the initial rule for the definition of the concept to be learned, which is **generalized** at the encounter of another positive example that is not covered
    - principle: dropping features from the rule
  - **Version-spaces search**: combines (1) and (2) until two hypotheses converge

Learning of rules and trees

How to reduce the search space while still obtaining a complete and consistent hypothesis?

1. Divide the search space into subspaces:
   - learn a rule for each subspace

     e.g., by searching a rule that covers most of the positive examples and removal of these examples from further training

2. Prefer simple rules above complex ones

Learning of rules and trees

3. Restrict the search space:
   - by considering a single best feature for inclusion or exclusion at each stage of building a rule

     = **greedy algorithm**: because backtracking is not used, i.e., each new choice depends on the previous choices, a good but not always optimal hypothesis is obtained
   - by using heuristics (branch and bound)

Learning of rules and trees

- Trees (cf. rules): often general-to-specific (top-down construction): e.g. C4.5 (Quinlan, 1993)
  - the feature that alone best (e.g., highest information gain) classifies the training examples is selected and used as the test at the root node of the tree
  - a descendant of the root node is then created for each possible value of this feature
  - training examples are sorted to the appropriate descendant node
  - the process is repeated using the training examples associated with each descendant node in order to select the best feature to test at this point of the tree
  - greedy algorithm

Learning of rules and trees

- To avoid overfitting:
  - growing covering rules or trees
  - pruning of overfit rules or trees: when the pruned rules or tree performs no worse than the original over a validation or test set
- Class assignment to a new text:
  - a rule that is evaluated as true is an indication of its class
  - a tree: starting at the root of the tree and moving through it (evaluating each node) until a leaf (class of the object) is encountered

Learning of rules and trees

- Advantages:
  - easily supplemented with handcrafted knowledge or verified
  - technique useful when conditional dependencies among features are present (often in texts!)
- Disadvantage:
  - text categorization: many features => greedy search of hypothesis space

Naive Bayes (NB) model

- Bayesian classifier:
  - the posterior probability that a new, previously unseen object belongs to a certain class given the features of the object is computed:
    - based on the probabilities that these individual features are related to the class

- **Naive** Bayes classifier:
  - computations simplified by the assumption that the features are conditionally independent

Naive Bayes model

$$P(C_j|w_1,\ldots,w_p) = \frac{P(w_1,\ldots,w_p|C_j)P(C_j)}{P(w_1,\ldots,w_p)}$$

where $w_1,\ldots, w_p$ = set of *p* features

ranking

$$P(C_j|w_1,\ldots,w_p) = P(w_1,\ldots,w_p|C_j)P(C_j)$$

independence assumption

$$= P(C_j)\prod_{i=1}^{p} P(w_i|C_j)$$

practical implementation

$$= \log P(C_j) + \sum_{i=1}^{p} \log P(w_i|C_j)$$

Naive Bayes model

normalized form

$$P(C_j|w_1,\ldots,w_p) = P(C_j)\frac{P(C_j)\prod_{i=1}^{p} P(w_i|C_j)}{\sum_{k=1}^{|C|} P(C_k)\prod_{i=1}^{p} P(w_i|C_k)}$$

Naive Bayes model

- Estimations from training set:
  - $P(C_j)$
  - $P(w_i|C_j)$:
    - **binomial or Bernoulli model**:
      - fraction of objects of class C_j in which feature w_i occurs
    - **multinomial model**:
      - fraction of times that feature w_i occurs across all objects of class C_j
      - additional positional independence assumptions
  - To avoid zero probabilities: add one to each count (Laplace smoothing)

```
TRAINBERNOULLINB(C, D)
1  V ← EXTRACTVOCABULARY(D)
2  N ← COUNTDOCS(D)
3  for each c ∈ C
4  do N_c ← COUNTDOCSINCLASS(D, c)
5     prior[c] ← N_c / N
6     for each t ∈ V
7     do N_ct ← COUNTDOCSINCLASSCONTAININGTERM(D, c, t)
8        condprob[t][c] ← (N_ct + 1) / (N_c + 2)
9  return V, prior, condprob

APPLYBERNOULLINB(C, V, prior, condprob, d)
1  V_d ← EXTRACTTERMSFROMDOC(V, d)
2  for each c ∈ C
3  do score[c] ← log prior[c]
4     for each t ∈ V
5     do if t ∈ V_d
6           then score[c] += log condprob[t][c]
7           else score[c] += log(1 − condprob[t][c])
8  return arg max_{c∈C} score[c]
```

Naive Bayes algorithm (binomial or Bernoulli model): training and testing.

[Manning et al. 2009]

TRAINMULTINOMIALNB($\mathbb{C}$, $\mathbb{D}$)

1 $V \leftarrow$ EXTRACTVOCABULARY($\mathbb{D}$)
2 $N \leftarrow$ COUNTDOCS($\mathbb{D}$)
3 **for each** $c \in \mathbb{C}$
4 **do** $N_c \leftarrow$ COUNTDOCSINCLASS($\mathbb{D}, c$)
5 $prior[c] \leftarrow N_c/N$
6 $text_c \leftarrow$ CONCATENATETEXTOFALLDOCSINCLASS($\mathbb{D}, c$)
7 **for each** $t \in V$
8 **do** $T_{ct} \leftarrow$ COUNTTOKENSOFTERM($text_c, t$)
9 **for each** $t \in V$
10 **do** $condprob[t][c] \leftarrow \frac{T_{ct}+1}{\sum_{t'}(T_{ct'}+1)}$
11 **return** $V, prior, condprob$

APPLYMULTINOMIALNB($\mathbb{C}, V, prior, condprob, d$)

1 $W \leftarrow$ EXTRACTTOKENSFROMDOC(V, d)
2 **for each** $c \in \mathbb{C}$
3 **do** $score[c] \leftarrow \log prior[c]$
4 **for each** $t \in W$
5 **do** $score[c] \mathrel{+}= \log condprob[t][c]$
6 **return** $\arg\max_{c \in \mathbb{C}} score[c]$

Figure 13.2 Naive Bayes algorithm (multinomial model): Training and testing.

[Manning et al. 2009]

Table 13.3 Multinomial versus Bernoulli model.

	multinomial model	Bernoulli model
event model	generation of token	generation of document
random variable(s)	$X = t$ iff t occurs at given pos	$U_t = 1$ iff t occurs in doc
document representation	$d = \langle t_1, \ldots, t_k, \ldots, t_{n_d} \rangle$, $t_k \in V$	$d = \langle e_1, \ldots, e_i, \ldots, e_M \rangle$, $e_i \in \{0, 1\}$
parameter estimation	$\hat{P}(X = t\|c)$	$\hat{P}(U_i = e\|c)$
decision rule: maximize	$\hat{P}(c) \prod_{1 \le k \le n_d} \hat{P}(X = t_k\|c)$	$\hat{P}(c) \prod_{t_i \in V} \hat{P}(U_i = e_i\|c)$
multiple occurrences	taken into account	ignored
length of docs	can handle longer docs	works best for short docs
# features	can handle more	works best with fewer
estimate for term the	$\hat{P}(X = \text{the}\|c) \approx 0.05$	$\hat{P}(U_{\text{the}} = 1\|c) \approx 1.0$

[Manning et al. 2009]

Naive Bayes model

- Class assignment:
  - find the *k* most probable classes; *k* = 1: $\arg\max_{C_j} P\left(C_j \middle| w_1,\ldots,w_p\right)$
  - alternative: select classes for which $P\left(C_j \middle| w_1,\ldots,w_p\right) > \text{threshold}$
- Advantages:
  - Efficiency
- Disadvantages:
  - Independence assumptions
  - When classifying text based on unigrams: no accurate probability estimates: close to 0; winning class after normalization close to 1

Support vector machine

- **Support vector machine**:
  - when two classes are linearly separable:
    - find a hyperplane in the *p*-dimensional feature space that best separates with **maximum margins** the positive and negative examples
    - maximum margins: with maximum Euclidean distance (= margin *d*) to the closest training examples (**support vectors**)
    - e.g., decision surface in two dimensions
  - idea can be generalized to examples that are not necessarily linearly separable and to examples that cannot be represented by linear decision surfaces

Support vector machine

- **Linear support vector machine**:
  - case: **trained on data that are separable** (simple case)
  - input is a set of *n* training examples:

$$S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$$

where $\mathbf{x}_i \in \Re^p$ and $y_i \in \{-1,+1\}$ indicating that $\mathbf{x}_i$ is a negative or positive example respectively

Support vector machine

Suppose we have some hyperplane which separates the positive from the negative examples, the points which lie on the hyperplane satisfy:

$$\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b = 0$$

where $\mathbf{w}$ = normal to the hyperplane

$\frac{|b|}{\|\mathbf{w}\|}$ = perpendicular distance from the hyperplane to the origin

$\|\mathbf{w}\|$ = Euclidean norm of $\mathbf{w}$

let d_+ (d_-) be the shortest distance from the separating hyperplane to the closest positive (negative) example

define the margin of the separating hyperplane to be d_+ and d_-

search the hyperplane with largest margin

Given separable training data that satisfy the following constraints:

$$\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \geq +1 \qquad \text{for } y_i = +1 \qquad (1)$$

$$\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \leq -1 \qquad \text{for } y_i = -1 \qquad (2)$$

which can be combined in 1 set of inequalities:

$$y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 \geq 0 \qquad \text{for } i = 1, \ldots, n \qquad (3)$$

The hyperplane that defines one margin is defined by:

$$H_1 : \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b = 1$$

with normal $\boldsymbol{w}$ and perpendicular distance from the origin

$$\frac{|1 - b|}{\|\mathbf{w}\|}$$

The hyperplane that defines the other margin is defined by:

$$H_2 : \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b = -1$$

with normal $\boldsymbol{w}$ and perpendicular distance from the origin

$$\frac{|-1 - b|}{\|\mathbf{w}\|}$$

Hence $d_+ = d_- = \frac{1}{\|\mathbf{w}\|}$ and the margin = $\frac{2}{\|\mathbf{w}\|}$

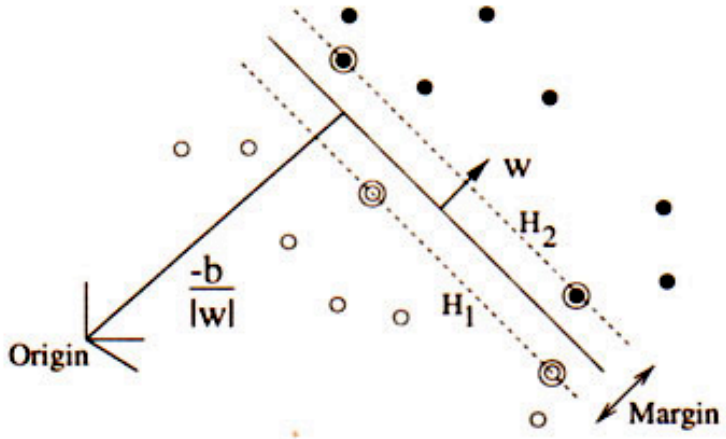


Figure 5. Linear separating hyperplanes for the separable case. The support vectors are circled.

[Burges 1998]

Hence we assume the following objective function to maximize the margin:

$$\text{Minimize}_{w,b} \ \langle \mathbf{w} \cdot \mathbf{w} \rangle$$

$$\text{Subject to } y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 \geq 0, \quad i = 1,\ldots,n$$

A dual representation is obtained by introducing Lagrange multipliers λ_i, which turns out to be easier to solve:

$$\text{Maximize } W(\lambda) = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i,j=1}^{n} \lambda_i \lambda_j y_i y_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle \qquad (4)$$

$$\text{Subject to: } \lambda_i \geq 0$$

$$\sum_{i=1}^{n} \lambda_i y_i = 0 \ , \quad i = 1,\ldots,n$$

Yielding the following decision function:

$$h(\mathbf{x}) = sign(f(\mathbf{x}))$$

$$f(\mathbf{x}) = \sum_{i=1}^{n} \lambda_i y_i \langle \mathbf{x}_i \cdot \mathbf{x} \rangle + b \qquad (5)$$

The decision function only depends on support vectors, i.e., for which $\lambda_i > 0$. Training examples that are not support vectors have no influence on the decision function

Support vector machine

- **Trained on data not necessarily linearly separable (soft margin SVM)**:
  - the amount of training error is measured using slack variables ξ_i the sum of which must not exceed some upper bound
  - The hyperplanes that define the margins are now defined as:

$$H_1 : \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b = 1 - \xi_i$$

$$H_2 : \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b = -1 + \xi_i$$

- Hence we assume the following objective function to maximize the margin:

$$\text{Minimize}_{\xi, \boldsymbol{w}, b} \left\langle \mathbf{w} \cdot \mathbf{w} \right\rangle + C \sum_{i=1}^{n} \xi_i^2$$

$$\text{Subject to } y_i(\left\langle \mathbf{w} \cdot \mathbf{x}_i \right\rangle + b) - 1 + \xi_i \geq 0 \ , \quad i = 1,...,n$$

where $\sum_{i=1}^{n} \xi_i^2$ = penalty for misclassification

C = weighting factor

The decision function is computed as in the case of data objects that are linearly separable (cf. (5))

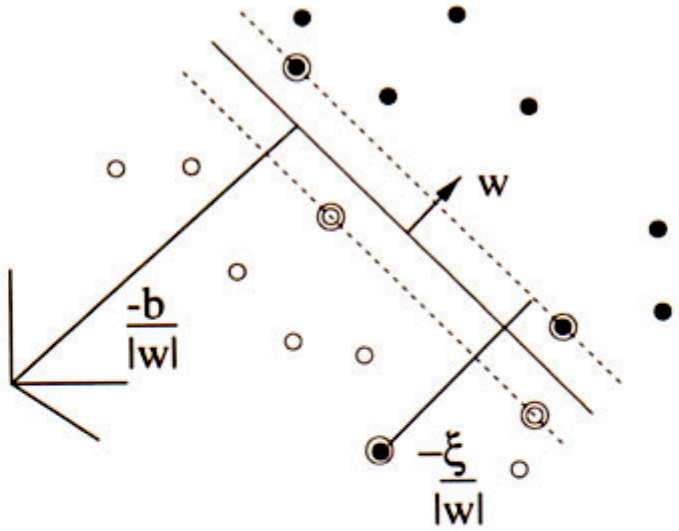


Figure 6. Linear separating hyperplanes for the non-separable case.

[Burges DMKD 1998]

Support vector machine

- When classifying natural language data, it is not always possible to linearly separate the data: in this case we can map them into a feature space where they are linearly separable
- Working in a high dimensional feature space gives computational problems, as one has to work with very large vectors
- In the dual representation the data appear only inside inner products (both in the training algorithm shown by (4) and in the decision function of (5)): in both cases a kernel function can be used in the computations

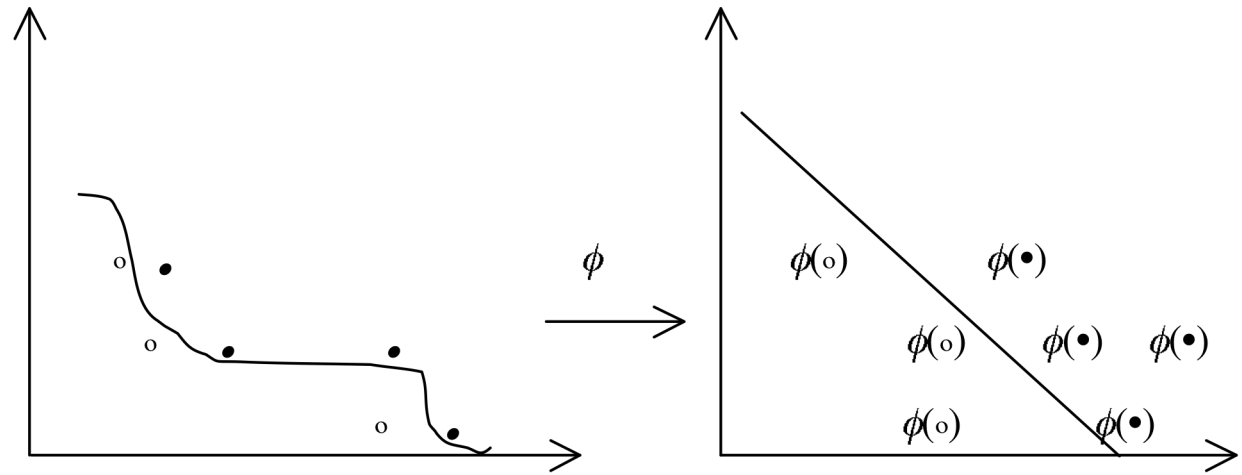


Fig. 5.2. A mapping of the features can make the classification task more easy (after Christianini and Shawe-Taylor 2000).

Kernel function

- A kernel function K is a mapping K: S x $S \rightarrow [0, \infty]$ from the instance space of examples S to a similarity score:

$$K(\mathbf{x}_i,\mathbf{x}_j) = \langle \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) \rangle$$

- In other words a kernel function is an inner product in some feature space
- The kernel function must be:
  - symmetric $[K(\mathbf{x}_i,\mathbf{x}_j) = K(\mathbf{x}_j,\mathbf{x}_i)]$
  - positive semi-definite: if $\mathbf{x}_1,\ldots,\mathbf{x}_n \in S$, then the n x n matrix G (*Gram matrix or kernel matrix*) defined by $G_{ij} = K(\mathbf{x}_i,\mathbf{x}_j)$ is positive semi-definite*

* has non-negative eigenvalues

Support vector machine

- Typical kernel functions: linear (mostly used in text categorization), polynomial, radial basis function (RBF)
- We can define kernel functions that (efficiently) compare strings (**string kernel**) or trees (**tree kernel**)
- The decision function $f(\mathbf{x})$ we can just replace the dot products with kernels $K(\mathbf{x}_i,\mathbf{x}_j)$:

$$h(\mathbf{x}) = sign(f(\mathbf{x}))$$

$$f(\mathbf{x}) = \sum_{i=1}^{n} \lambda_i y_i \langle \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}) \rangle + b$$

$$f(\mathbf{x}) = \sum_{i=1}^{n} \lambda_i y_i K(\mathbf{x}_i,\mathbf{x}) + b$$

- For example, the similarity between sentences can be defined by a tree kernel, where the nodes in the tree have attributes attached: similarity computation takes into account the structure of the parse trees and the similarity between attributes in nodes

Support vector machine

- Advantages:
  - SVS can cope with many (noisy) features: no need for a priori feature selection, though you might select features for reasons of efficiency
  - many text categorization problems are linearly separable

Text categorization results

- Symbolic techniques:
  - good results in 90% recall and precision but only tested in limited domains
- Machine learning techniques:
  - current results see below
  - **Benchmark collection** for text categorization evaluation: Reuters-21578 : « easy » collection ?

Example of a Reuters news wire

BC-FRESH-DEBATE-ON-NORWAY 06-02 0091 OSLO, June 2 - European Community officials welcomed Norway's recent move to renew a public debate on Community membership but said Norway should not expect special trade advantages as long as it stays outside the EC. Belgian Willy de Clercq, EC Commissioner on external affairs and trade policy, said high level talks this week with Norway's minority Labour government had helped clarify several misconceptions that led to Norway's narrow rejection of EC membership in a 1972 referendum. "But you (Norway) cannot be in the club and remain outside the club. You can expect equal footing in the club, but not out of it," de Clercq added, referring to Norway's attempts to adapt its EC trade ties in the face of Community moves to launch an internal trade market from 1992. The government, worried that the internal market will hamper trade with the EC, which takes about two-thirds of Norway's exports, last month sent a report to parliament asking political parties and the public to reassess the country's relationship to the EC.

Categories: norway de-clercq ec

Text categorization results

- Experiments of [Li & Yamanishi IP & M 2002]: Reuters-21578:
  - 90 categories
  - “ModApte split”: training set: 9603 documents; test set: 3299 documents
  - vocabulary: 21,995 unique words (bodies of texts used); 8611 unique words (titles of texts used)
  - average number of words per text: 70.2 (bodies of text used); 6.4 (titles of texts used)

Micro-averaged breakeven points for methods in % of Reuters-21578:

	Bodies of texts used	Titles of texts used
Decision rules: DL-ESC	82.0	78.5
Decision rules: DL-SC	78.3	78.0
Decision trees: C4.5	79.9	77.8
NB	77.3	73.6
SVM	84.1	77.3

DL-ESC: feature selection based on principle of Stochastic Complexity (SC) which is an improved variant of information gain; growing and pruning of rules: simple rules; principle of minimizing Extended Stochastic Complexity (ESC) which minimizes the classification error taking into account the sample size
DL-SC: feature selection, growing and pruning of rule set based on on principle of Stochastic Complexity (SC)

[Li & Yamanishi IP & M 2002]

Text categorization results

- Experiments of [Joachims 2002]: Reuters-21578
  - selected set of categories
  - vocabulary: 27,658 unique words
  - *tf* x *idf* weighting

					SVM (poly) degree d =					SVM (rbf) width γ =			
	Bayes	Rocchio	C4.5	k-NN	1	2	3	4	5	0.6	0.8	1.0	1.2
earn	95.9	96.1	96.1	97.3	98.2	98.4	**98.5**	98.4	98.3	**98.5**	98.5	98.4	98.3
acq	91.5	92.1	85.3	92.0	92.6	94.6	**95.2**	95.2	95.3	95.0	95.3	95.3	**95.4**
money-fx	62.9	67.6	69.4	78.2	66.9	72.5	75.4	74.9	**76.2**	74.0	75.4	**76.3**	75.9
grain	72.5	79.5	89.1	82.2	91.3	93.1	**92.4**	91.3	89.9	**93.1**	91.9	91.9	90.6
crude	81.0	81.5	75.5	85.7	86.0	87.3	88.6	**88.9**	87.8	**88.9**	89.0	88.9	88.2
trade	50.0	77.4	59.2	77.4	69.2	75.5	76.6	77.3	**77.1**	76.9	78.0	**77.8**	76.8
interest	58.0	72.5	49.1	74.0	69.8	63.3	67.9	73.1	**76.2**	74.4	75.0	**76.2**	76.1
ship	78.7	83.1	80.9	79.2	82.0	85.4	86.0	**86.5**	86.0	**85.4**	86.5	87.6	87.1
wheat	60.6	79.4	85.5	76.6	83.1	84.5	85.2	**85.9**	83.8	**85.2**	85.9	85.9	85.9
corn	47.3	62.2	87.7	77.9	86.0	86.5	85.3	**85.7**	83.9	**85.1**	85.7	85.7	84.5
microavg.	**72.0**	**79.9**	**79.4**	**82.3**	84.2	85.1	85.9	86.2	85.9	86.4	86.5	86.3	86.2
					combined: **86.0**					combined: **86.4**			

Fig. 2. Precision/recall-breakeven point on the ten most frequent Reuters categories and microaveraged performance over all Reuters categories. k-NN, Rocchio, and C4.5 achieve highest performance at 1000 features (with $k = 30$ for k-NN and $\beta = 1.0$ for Rocchio). Naive Bayes performs best using all features.

[Joachims 2002]

Text categorization results

- Experiments of [Qian et al. JIS 2007]: Reuters-21578
  - selected set of categories
  - “ModApte split”: training set: 9603 documents; test set: 3299 documents
  - vocabulary: 21,578 unique words
  - HARMONY, SAT-MOD, ARC-BC: frequent item set patterns and rule induction
  - ATC-HPs: hyperclique patterns and rule induction

Table 8
A Comparison Result on the *Reuters* Data Set (Parameters: minsup=0.04, τ=0.60, δ=0.35, K=280).

category	ATC -HPs	ARC -BC	HAR MONY	SAT -MOD	Naïve Bayes	Bayes Net	Decision Trees	Linear SVM
acq	96.5	90.9	95.3	95.1	87.8	88.3	89.7	93.6
corn	86.2	69.6	78.2	71.2	65.3	76.4	91.8	90.3
crude	87.4	77.9	85.7	90.6	79.5	79.6	85.0	88.9
earn	96.6	92.8	98.1	97.4	95.9	95.8	97.8	98.0
grain	85.8	68.8	91.8	91.3	78.8	81.4	85.0	94.6
interest	78.5	70.5	77.3	74.9	64.9	71.3	67.1	77.7
money-fx	84.4	70.5	80.5	86.6	56.6	58.8	66.2	74.5
ship	87.3	73.6	86.9	83.6	85.4	84.4	74.2	85.6
trade	90.1	68.0	88.4	84.9	63.9	69.0	72.5	75.9
wheat	82.3	84.8	62.8	75.2	69.7	82.7	92.5	91.8
micro-avg	**92.6**	82.1	92.0	92.2	81.5	85.0	88.4	92.0
macro-avg	**87.5**	76.7	**84.5**	**85.1**	74.8	78.8	82.2	**87.1**

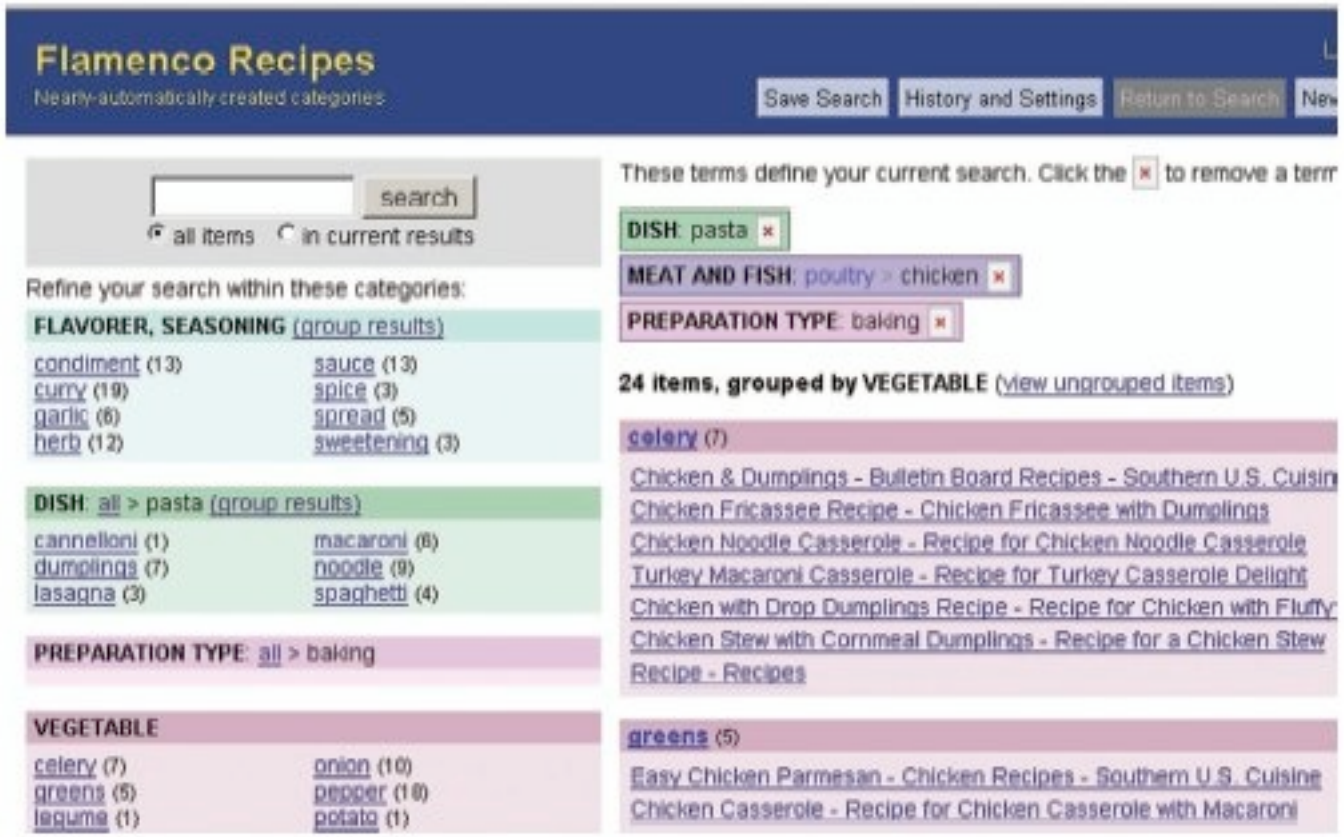
Values are micro-averaged and macro-averaged F_1

[Qian et al. JIS 2007]

Hierarchical categorization

- What kind of category structure is most effective for exploration and browsing of information collections?
- **Hierarchical faceted categories** = a set of category hierarchies, each of which corresponds to a different facet and contains meaningful concepts C_i relevant to the collection to be navigated

- Hierarchy can be exploited in text categorization, in order to improve
  - efficiency of the classification and training
  - accuracy of the classification



[Hearst CACM 2007]

Hierarchical categorization

- Often a divide-and-conquer strategy:
  - **Training**:
    - classifier is built independently at each internal node of the hierarchy using the documents of the subcategory of this hierarchy
    - feature selection is often performed at each node
  - **Testing**: application of tree of classifiers:
    - greedy: at each node the learned classifier directs the documents to a subtree until a leaf node or other stopping criterion is reached (e.g., probability of class is higher than a threshold)

Hierarchical categorization

- Traverse possible paths from root to leave and compute best path by e.g. combining classification probabilities at each node in product
- Traverse subset of paths by selecting branches for with the classification probability > threshold

- Advantage of this model:
  - When adding categories in the taxonomy, not all classifiers need to be updated

Hierarchical categorization: problems

- **Large-scale taxonomies** (e.g., Web):
  - greedy class assignment: error made high in the hierarchy cannot be recovered
  - higher nodes demand complex decision surfaces
  - increasing sparsity at deeper nodes in the hierarchy (not treated here)

Hierarchical categorization: solutions

- **Reducing error propagation**:
  - biasing the training distribution by training on the same type of distribution as will be seen during testing
  - given ancestor nodes are trained: use as training data at a node: the actual training data augmented with the predicted distribution

    (called Refinement in results below)

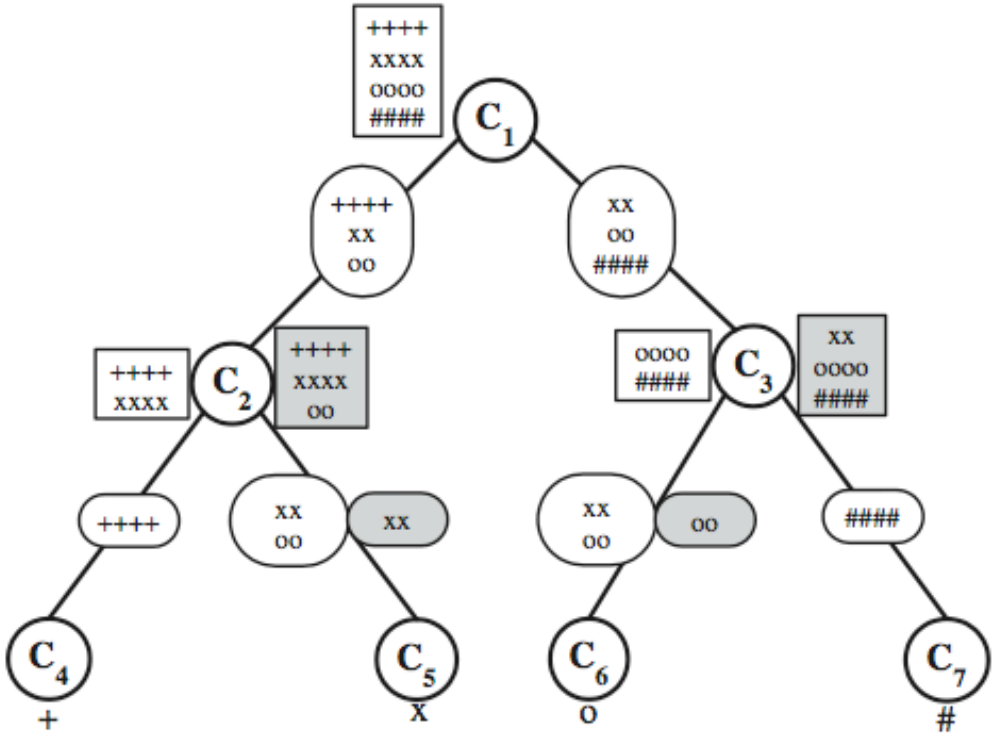


Figure 1: Refinement uses predicted behavior to change the training distribution and learns models that filter out errors. The *training* set for the standard approach is illustrated with white boxes and in gray boxes when Refinement would differ. The effect on a *test* set is illustrated with standard classifications in white ovals and in gray when Refinement would differ.

[Bennett &Nguyen SIGIR 2009]

Hierarchical categorization: solutions

- **Complex decision surfaces** in higher nodes:
  - Not often captured with basic linear classifiers and non-linear classifiers might lead to overfitting
  - Including membership in lower-level categories as a feature: rough guess at the lower level classification and propagation of this information before top-down classification
  - Classification information from children and their cousins trained with all positive data for the node and all documents not belonging to the node's parents as negative data

  (+ Refinement Model => called Refined Experts in the results below)

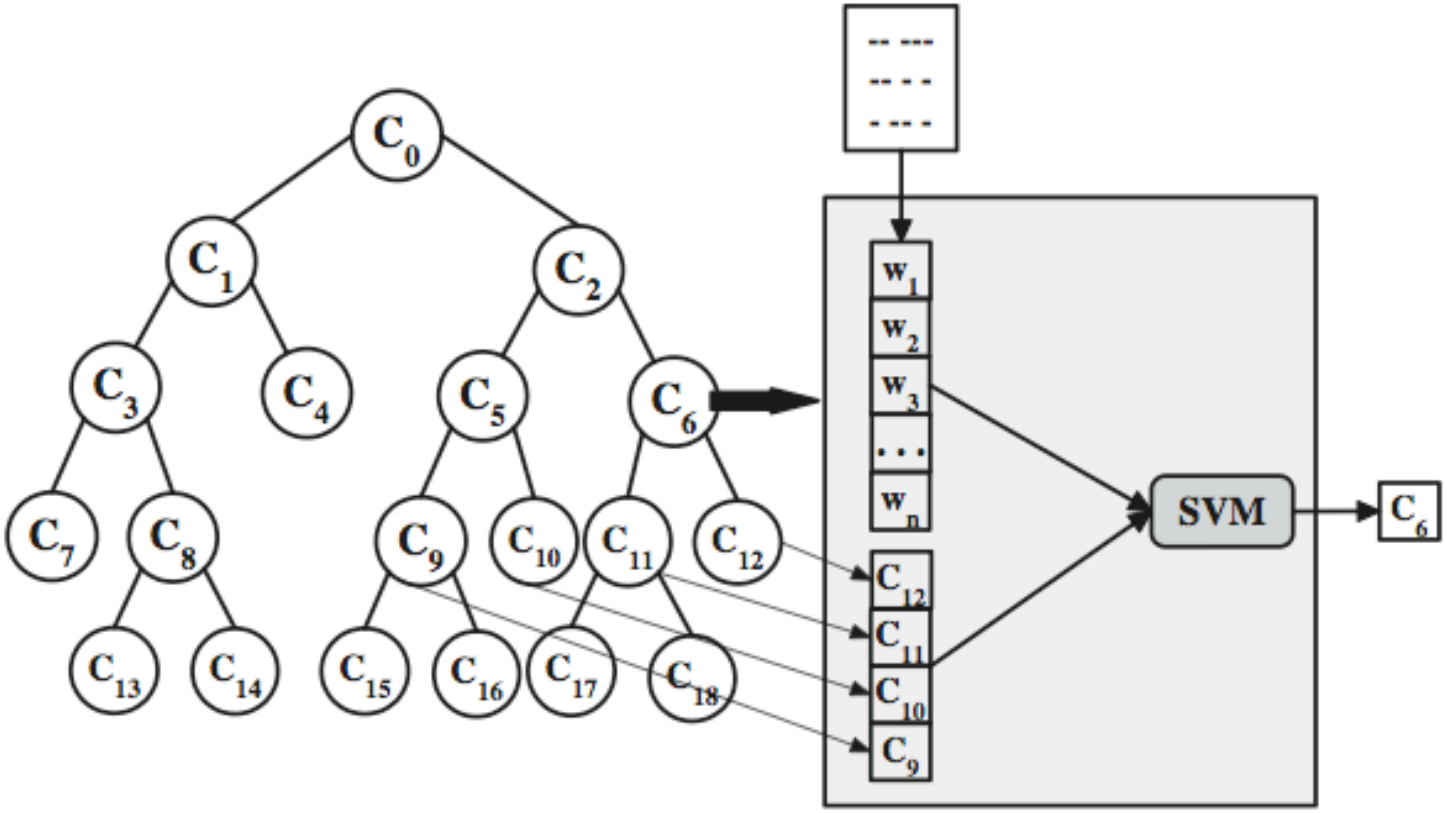


Figure 2: Refined Experts augments the representation of a document with a set of metafeatures containing membership predictions from lower nodes (here its children and their "cousins").

[Bennett & Nguyen SIGIR 2009]

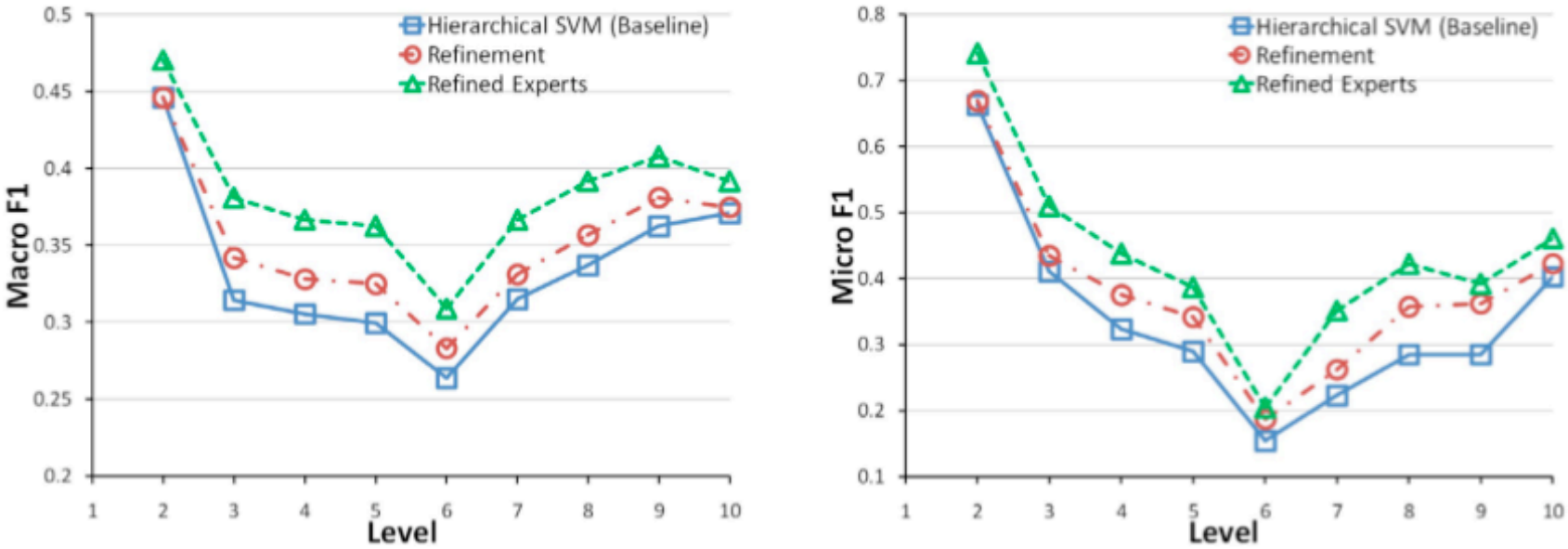


Figure 5: Macro (*left*) and Micro (*right*) F1 broken down by level of the hierarchy.

Web pages from the Open Directory Project's hierarchy of the Web, except World and Regional branches, content crawled in January 2008.

[Bennett & Nguyen SIGIR 2009]

What have we learned?

- Text categorization: use mainly techniques of supervised learning:
  - for “easy”, large training data: the effectiveness of the learned model approaches the one of human categorization
  - most successful:
    - **for long texts: SVM**
    - **for short texts: decision rules and trees**
    - **efficiency and acceptable results: MNB**
- Hierarchical classification: knowledge of the hierarchy of the classes can be exploited in a variety of ways

Research questions to be solved

- More research is needed:
  - to learn from few training examples (to cope with changing document collections and classifications):
    - feature selection = important: to select the right features: potential of NLP, information extraction, pattern discovery
    - feature transfer
  - to learn concepts of small text passages
  - to better exploit relationships between categories (e.g., hierarchical) in learning

Further reading

Bennett, P.N. & Nguyen, N. (2009). Refined experts: Improving classification in large taxonomies. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 11-18). New York: ACM.

Burges, C.J.C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery,* 2, 121-167.

Cheng, H., Yan, X., Han, J. & Hsu, C.-W (2007). Discriminative frequent pattern analysis for effective classification. In *Proceedings of the IEEE 23rd International Conference on Data Engineering* (pp. 716-725). IEEE Computer Society Press.

Christianini, N. & Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines.* Cambridge, UK: Cambridge University Press.

Hearst, M.A. (2006). Clustering versus faceted categories for information extraction. *Communications of the ACM*, 49 (4), 59-61.

Joachims, T. (2001). A statistical learning model of text classification for support vector machines. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 128-136). New York: ACM.

Joachims, T. (2002). *Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms.* Boston, MA: Kluwer Academic Publishers.

Li, H. & Yamanishi, K. (2002). Text classification using ESC-based stochastic decision lists. *Information Processing & Management,* 38 (3), 343-361.

Manning,C.D., Raghavan, P. & Schütze, H. (2009). *An Introduction to Information Retrieval.* Cambridge University Press Cambridge, England. http://nlp.stanford.edu/IR-book/information-retrieval-book.html

Qian, T., Xiong, H., Wang, Y. & Chen E. (2007). On the strength of hyperclique patterns for text categorization. *Journal of Information Science,* 177 (19), 4040-4058.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1), 1-47.

Yang, Y., & Liu, X. (1999). A re-examination of text categorization methods. In *Proceedings of the 22nd International Conference on Research and Development in Information Retrieval* (pp. 42-49). ACM: New York.

Yang, Y. & Pedersen, J. (1997). A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning* (pp. 412-420). San Francisco, CA: Morgan Kaufman.

Xiong, H., Tan, P. & Kumar, V. (2006). Hyperclique pattern discovery. *Data Mining and Knowledge Discovery,* 13 (2), 219-242.

Zhang, J. & Yang, Y. (2003). Robustness of regularized linear classification methods in text categorization. In *Proceedings of the Twenty-Sixth Annual International ACM SIGIR Conference in Information Retrieval* (pp. 190-197). ACM: New York.