Yiming Yang, 15-381 AI, Tuesday, Oct 24, 2000

Decision Trees in Text Categorization

Example: shopping on the Web

Suppose you want to buy a cappucino maker as a gift

– try Google for "shopping cappucino maker"

– try "Yahoo! Shopping" for "cappucino (or coffee) maker"

Observations:

- Broad indexing & speedy search alone are not enough
- Organizational view of data is critical for effective retrieval
- Categorized data (Yahoo!) are easy for user to browse
- Category taxonomies become most central in well-known web sites (Yahoo!, Lycos, AltaVista, InfoSeek, WebCrawler ...)

Related Questions:

- *What is the cost of category-based indexing and retrieval?*
- *Techniques available for automated text categorization?*

Task Definition

TC: assign predefined categories to documents (and queries)

Applications

- organizing web pages using category hierarchies
- mapping queries to relevant categories and taxonomies
- indexing literature articles using subject categories (e.g., by the Library of Congress, MEDLINE, etc.)
- routing email messages to relevant user groups
- tracking news events about particular topics (e.g., MS trials)
- classifying responses to Census Bureau by occupations

...

Benefit

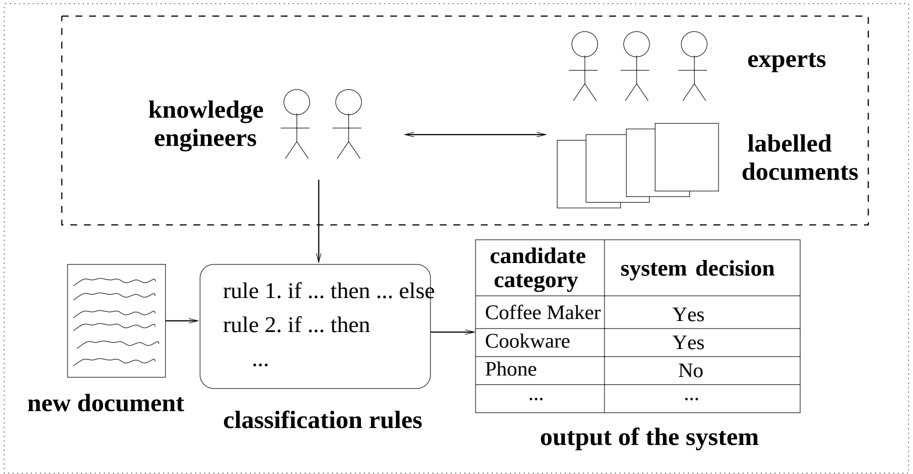
– providing organizational view of data

– allowing user to cut-off irrelevant parts and focus on must relevant part

Cost of Manual Text Categorization

- *Yahoo!*
  - About 200 people for manual labelling of Web pages
- Articles in MEDLINE
  - Medical Subject Headings (18,000 categories)
  - \$2 millions per year for manual indexing at NLM
- Patient-record event classification
  - International Classification of Diseases (ICD) for billing
  - \$1.4 million per year for manual coding at Mayo
- U.S. Census Bureau decennial census (1990: 22 million responses)
  - 232 industry categories and 504 occupation categories
  - \$15 millions if fully done by hand
- Patents over the world
  - International Patent Classification, 60,000 categories

Expert system for text categorization (late 1980s)



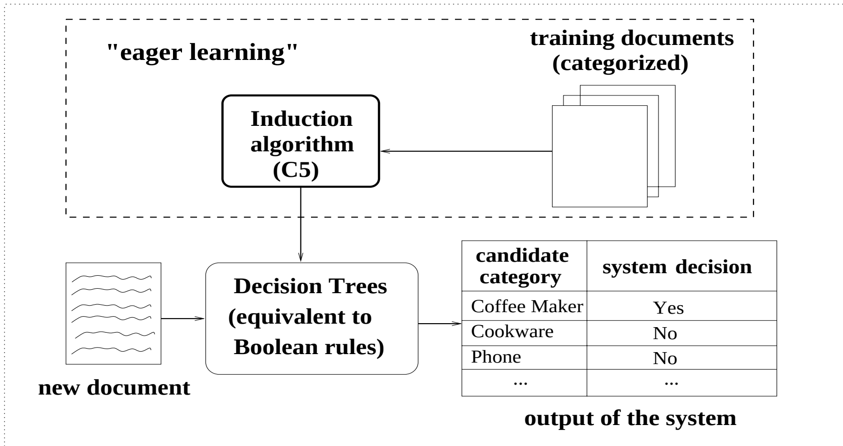
A document in category *Coffee Maker*:

"Saeco revolutionized *espresso* brewing a decade ago by introducing Saeco SuperAutomatic *machines*, which go from bean to *coffee* at the touch of a button. The all-new Saeco Vienna SuperAutomatic home coffee and *cappucino machine* combines top quality with low price!"

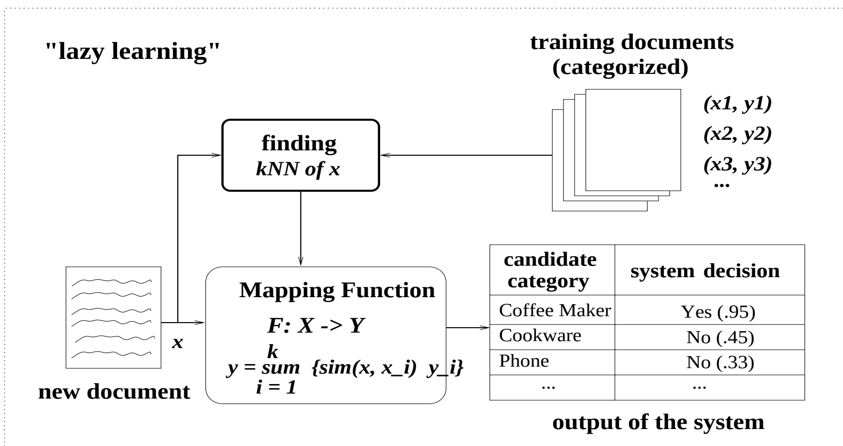
<u>Rule</u> 1. (*espresso* $\vee$ *coffee* $\vee$ *cappucino)* $\wedge$ machine* $\Rightarrow$ *Coffee Maker*

<u>Rule</u> 5. *automat** $\wedge$ *answering* $\wedge$ *machine** $\Rightarrow$ *Phone*

DTree induction for text categorization (since 1994)



kNN approach to text categorization (since 1992)



Knowledge Engineering vs Statistical Learning

For U.S. Census Bureau Decennial Census 1990

- 232 industry categories and 504 occupation categories
- \$15 millions if fully done by hand

Define classification rules manually:

- Expert System AIOCS
- Development time: 192 person-months (2 people, 8 years)
- Accuracy = 47%

Learn classification function algorithmicly:

- Nearest Neighbor classification (Creecy'92: 1-NN)
- Development time: 4 person-months (Thinking Machine)
- Accuracy = 60%

Approaches to Automated Text Categorization

- Regression based on Least Squares Fit (1991)
- Nearest Neighbor Classification (1992)
- Bayesian Probabilistic Models (1992)
- Symbolic Rule Induction (1994)
- Neural Networks (1995)
- Rocchio approach (traditional IR, 1996)
- Support Vector Machines (1997)
- Boosting or Bagging (1997)
- Hierarchical Language Modeling (1998)
- First-Order-Logic Rule Induction (1999)
- Maximum Entropy (1999)
- Hidden Markov Models (1999)
- Error-Correcting Output Coding (1999)

...

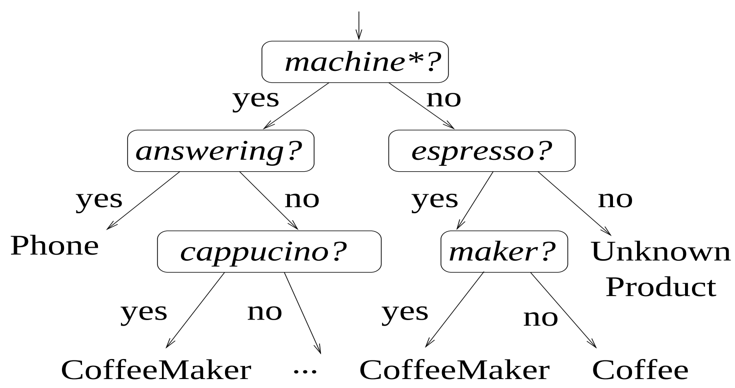
Results in TC benchmark evaluations

Text categorization methods on Reuters-21578 ApteMod Corpus
(about 10,000 documents in 90 categories)

METHOD	microF1	macroF1	Evaluated by
DTree+boost	.878	-	Apte'99 (IBM)
SVM	.860	.524	Joachims'98 (Dortmund)
kNN	.857	.513	Yang'94
LLSF	.855	.501	Yang'92
AdaBoost	.853	-	Schapire'99 (ATT)
Neural Nets	.841-.868	.344-.458	Yang'99
NaiveBayes	.796	.389	Yang'99
DTree C4.5	.789	-	Joachims'98 (Dortmund)
Rocchio	.787	-	Joachims'98 (Dortmund)

F_1: harmonic mean of recall and precision, $F_1 = 2rp/(r+p)$

Decision Tree implementing rules



Rules:

- **if** machine* & answering **then** Phone
- **if** machine* & (not answering) & cappucino **then** Coffee Maker
- **if** (not machine*) & expresso & (not maker) **then** Coffee
- ...

How to design the tree or the rules? By hand?

– too hard

– too much labor

– not optimal

DTree Induction

– Using a training set of categorized documents

– Producing one decision tree per category

An incomplete algorithm:

For each category in the training set, induce a tree as below

– replace the category labels with Yes or No in the training set

– assign the relabelled training set to the root node

– grow the tree recursively by calling *Grow(root)*

Grow (Node) – a recursive procedure taking a node as the input:

if some *stopping criterion* is met at the current node

then turn the current node into a *leaf*, i.e., label the node using the most common category among the training documents and terminate the precedure;

else

– choose the *most informative* word from the current training set

– devide the training documents into two subsets according to that word (occurs or not)

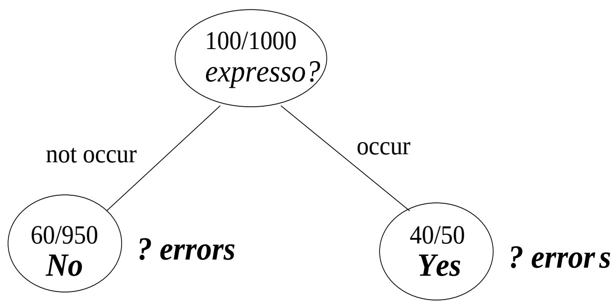
– create two children nodes (C_1 and C_2), each is with a subset

– call *Grow*(C_1) and *Grow*(C_2), respectively

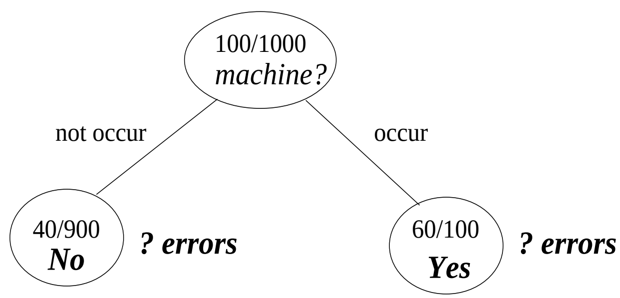
How to measure the informativeness of a word?

"Expresso" and "machine", which word is more informative wrt category "Coffee Maker"?

Given a collection of 1000 documents, 10% with YES

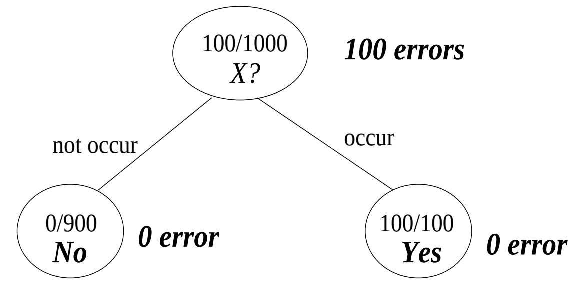


Given a collection of 1000 documents, 10% with YES

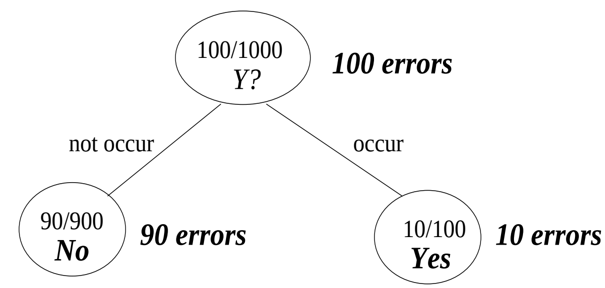


Two Extreme Cases

Maximum gain Total 1000 documents, 10% with YES



No gain Total 1000 documents, 10% with YES



Entropy of a document collection with 2 classes:

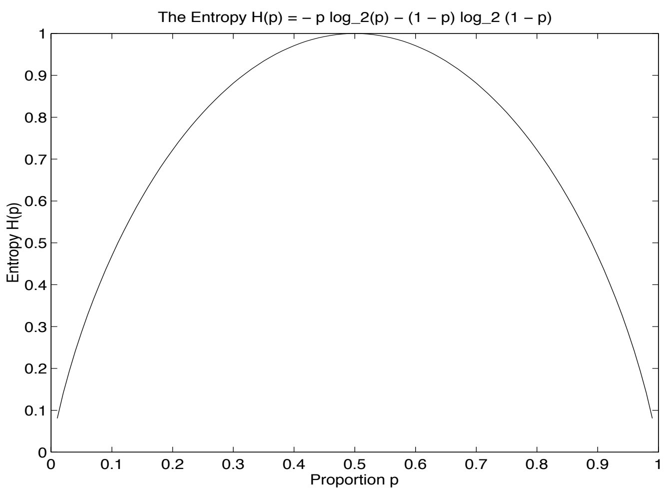
$$H(D) \stackrel{\text{def}}{=} -p \log_2 p - (1-p)\log_2(1-p)$$

where

D is a document collection;

p is the probability of a document in D belonging to YES;

$1-p$ is the probability of a document in D belonging to NO.



Entropy of a document collection with m classes

$$H(D) \stackrel{\text{def}}{=} \sum_{i=1}^{m} -p_i \log_2 p_i$$

where each document belongs to one and only one category; p_i is the probability of documents belonging to the ith category, and $\Sigma_{i=1}^{m} p_i = 1$.

Intuitions about Entropy $H(D)$:

- Entropy is a statistic about a collection of classified members (not about a word), a function of category distribution in that collection.
- According on Information Theory, it is the mininum number of bits needed for encoding the classification of an arbitrary member in the collection on average.[1]
- It reflects how difficult to predict the classification (YES or NO in 2-way classification) of an arbitrary member in the collection (given its distribution). For instance, it is most difficult to guess correctly when the chance is 50-50%, while it is equally easy when $p = 10\%$ or $p = 90\%$.
- It allows a comparison of a training set with its subsets when using a word to split the super set, to see how much the classification task is "eased".
- It provides a quantitative measure of how "informative" a word is, based on how well it splits a training set.

[1] By Shannon's Information Theory, the optimal coding is to use $\log_2 \frac{1}{p}$ bits per instance for the category with a probability of p.

Information Gain of term t in training set D:

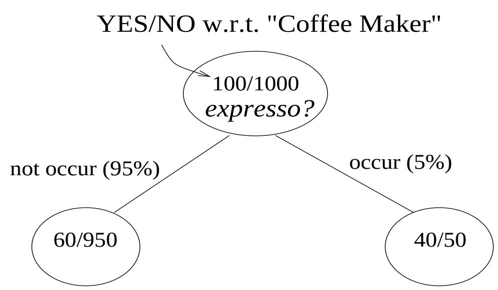
$$I(D,t) \stackrel{\text{def}}{=} H(D) - \{\frac{|D_t|}{|D|}H(D_t) + \frac{|D_{\bar{t}}|}{|D|}H(D_{\bar{t}})\}$$

where

D is a document collection;

$D_t \in D$ is the subset of documents containing t;

$D_{\bar{t}} \in D$ is the subset of documents not containing t.



$I(D, expresso) = \qquad -(.05 \times \qquad +.95\times \qquad) =$

$$H(_*) = -p\log_2 p - (1-p)\log_2(1-p)$$

$Probability$	D	D_t	$D_{\bar{t}}$
p	0.1	0.8	0.06
$1-p$	0.9	0.2	0.94
$p\log p$			
$(1-p)\log(1-p)$			
H			

Stopping criteria in DTree induction

If all the documents belong to the same category (or if they are sufficiently homogeneous);

if too few examples in the node (won't generalize on new examples); or

if all the terms have an IG value of zero.

Complete algorithm for growing a tree:

1. Set the root with the full training set as the *current node*.
2. **If** any of the stopping criteria is met at the current node

   then turn it into a *leaf* (or *terminal*) and label it with YES or NO using the most common category among the training documents;

   else

   - compute the information gain (IG) for each term in documents;
   - select the term with the largest IG value for the current node;
   - split the current node into two children nodes; and
   - recursively grow each child node using the same procedure.

Pros of DTree:

- Decision rules good in explaining why a category is relevant
- Easy to combine DTree rules with human-coded rules
- Fast online response ($O(mk)$) when the number of categories (m) is small

Cons of DTree:

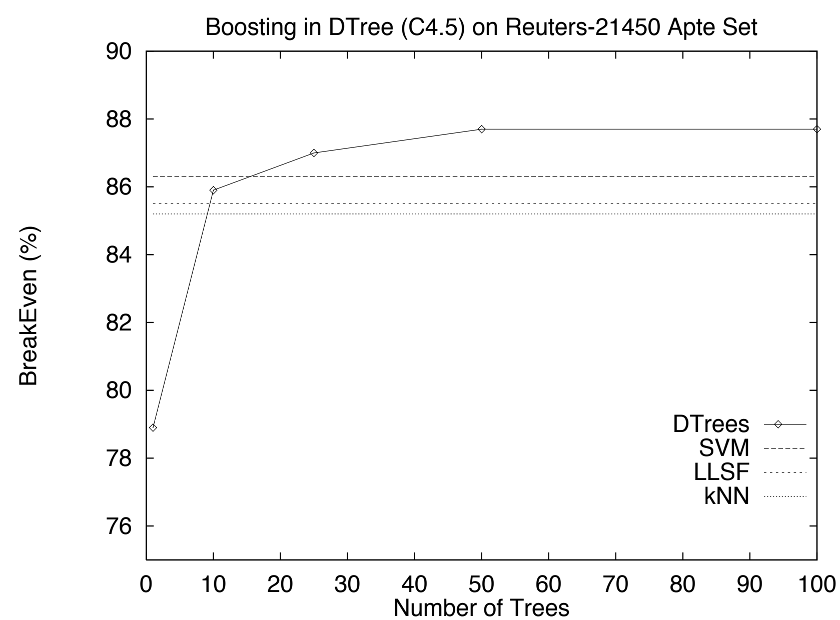
- Mediocre performance in TC benchmark evaluations, possibly due to
  - greedy algorithm, not optimal (choosing each word in isolation)
  - many terms would have same IG values (arbitrarily choose one)
  - not easy to use term weights (e.g., TF, IDF)
- Intensive training cost ($O(mkvn)$) especially when m is large

  m is the number of unique categories in the training set;
  k is the average depth (number of levels) of the resulting trees;
  v is the training-set vocabulary size; and
  n is the number of documents in the training set.
- Too costly when frequent re-training is needed

Improvement by Boosting (Apte et al. at IBM)

- Generate a forest of DTrees (e.g., 50) instead of one per category
- Adaptive re-sampling of training documents via iterations
- Duplicate the documents failed to classify
- Allow DTrees to vote (majority or weighted-majority vote)
- Best results on a benchmark collection in TC evaluations

  (improved from .79 to .88 in *Break-Even Point* (BEP))



Summary

- Data organization is necessary condition for effective retrieval
- Providing organizational view is critical for user to find relevant info
- Text categorization holds promise to both
- Many competing theories and algorithms
- Evaluation benchmarks begin to be established
- Some methods are scaled to very large applications, many are not
- Some methods are in daily use for computer-assisted human classification (e.g., kNN & LLSF at Mayo since 1994)
- Many more are under development for industrial applications (Internet, newswires, etc.)
- Scalability is a issue in vary large applications

References

Tom Mitchell. Machine Learning. *McCraw Hill*, 1996.

S.M. Weiss, C. Apte, F. Damerau, D.E. Johnson, F.J. Oles, T. Goets and T. Hampp". Maximizing Text-Mining Performance, *journal of IEEE Intelligent Systems, Special Issue on Applications of Intelligent Information Retrieval* 1999, vol. 14, No. 4, pp 63–69.

Y. Yang (1999). An evaluation of statistical approaches to text categorization, *Journal of Information Retrieval* volume 1, numbers 1/2, pages 67–88.

Performance Measures for TC Evaluation

Given n test documents and m classes in consideration, a classifier makes $n \times m$ binary decisions. For each class, a two-by-two contingency table is computed.

	truly YES	truly NO
system YES	a	b
system NO	c	d
	$a+b+c+d=n$	

Assuming $a+c>0$, $a+b>0$, and $b+d>0$:

- Recall $r = a/(a+c)$ where $a+c>0$;
- Precision $p = a/(a+b)$ where $a+b>0$;
- Miss $= c/(a+c) = 1-r$ where $a+c>0$;
- False alarm (fallout) $f = b/(b+d)$ where $b+d$;
- Accuracy $acc = (a+d)/n$;
- Error $err = (b+c)/n$;
- F-measure (assuming $r>0$ and $p>0$; otherwise, set $F_\beta(r,p)=0$):

$$F_\beta(r,p) = \frac{(\beta^2+1)pr}{\beta^2 p + r} \qquad \text{(weighted harmonic mean of } r \text{ and } p\text{)}$$

$$F_1 = 2rp/(r+p) \qquad \text{(harmonic mean of } r \text{ and } p\text{)}$$

- Break-even point $BEP = (r+p)/2$, the simple average of recall and precision.