Text Categorization

Is this spam?

From: "" <takworlld@hotmail.com>
Subject: real estate is the only way... gem oalvgkay

Anyone can buy real estate with no money down

Stop paying rent TODAY !

There is no need to spend hundreds or even thousands for similar courses

I am 22 years old and I have already purchased 6 properties using the methods outlined in this truly INCREDIBLE ebook.

Change your life NOW !

===

Click Below to order:
http://www.wholesaledaily.com/sales/nmd.htm
===

Categorization

- Given:
  - A description of an instance, $x \in X$, where X is the *instance language* or *instance space*.
  - A fixed set of categories: $C=\{c_1, c_2, \ldots c_n\}$
- Determine:
  - The category of x: $c(x) \in C$, where $c(x)$ is a categorization function whose domain is X and whose range is C.

Learning for Categorization

- A training example is an instance $x \in X$, paired with its correct category $c(x)$: $<x, c(x)>$ for an unknown categorization function, c.
- Given a set of training examples, D.
- Find a hypothesized categorization function, $h(x)$, such that:

$$\forall < x, c(x) > \in D : h(x) = c(x)$$

Consistency

Sample Category Learning Problem

- Instance language: <size, color, shape>
  - size ∈ {small, medium, large}
  - color ∈ {red, blue, green}
  - shape ∈ {square, circle, triangle}
- *C* = {positive, negative}
- *D*:

Example	Size	Color	Shape	Category
1	small	red	circle	positive
2	large	red	circle	positive
3	small	red	triangle	negative
4	large	blue	circle	negative

Another Example

- Predict stock market profits based on the age of the company, whether the company has competition, and the market sector):

Example	Age	Competition	Sector	Category
1	old	yes	software	down
2	old	no	hardware	down
3	new	yes	software	up
4	mid	no	hardware	up

General Learning Issues

- Many hypotheses are usually consistent with the training data.
- Bias
  - Any criteria other than consistency with the training data that is used to select a hypothesis.
- Classification accuracy (% of instances classified correctly).
  - Measured on independent test data.
- Training time (efficiency of training algorithm).
- Testing time (efficiency of subsequent classification).

Text Categorization

- Assigning documents to a fixed set of categories.
- Applications:
  – Web pages
    - Recommending
    - Yahoo-like classification
  – News articles
    - Personalized newspaper
  – Email messages
    - Routing
    - Prioritizing
    - Folderizing
    - spam filtering

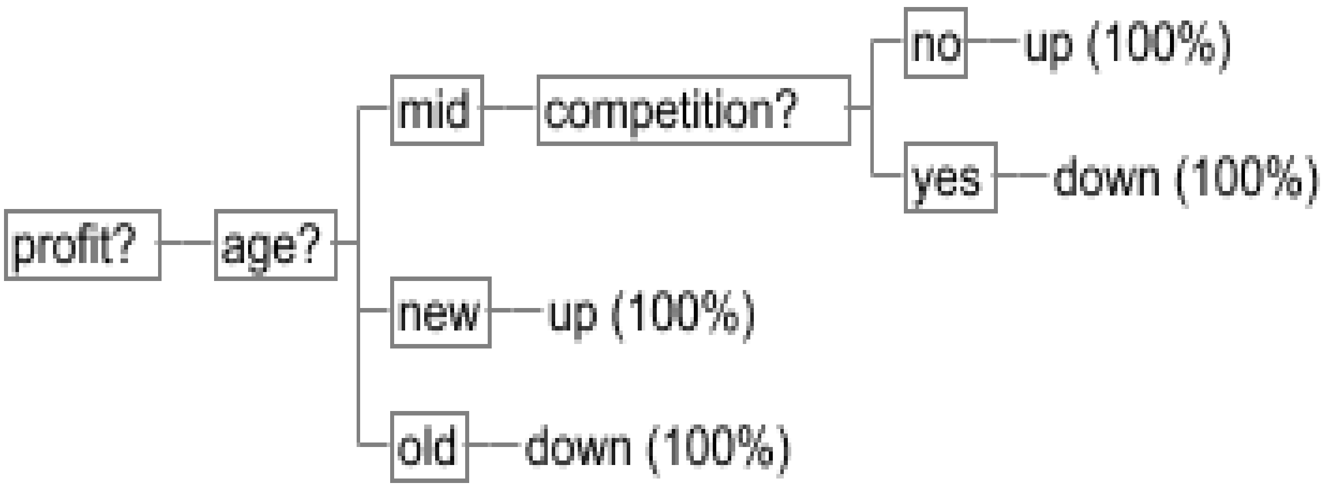
Learning for Text Categorization

- Manual development of text categorization functions is difficult.
- Machine Learning Algorithms:
  - **Decision Trees**
  - Naïve Bayes
  - Neural Networks
  - **Relevance Feedback (Rocchio)**
  - Rule based (Ripper)
  - **K Nearest Neighbor (case based)**
  - Support Vector Machines (SVM)

Decision Trees

- Information-gain algorithms for building decision tree from training data.
  - Greedy algorithm builds tree top down.
  - At each node, determine the test that "best" splits the remaining data.
  - "Best" split is the one that adds the most information.
- Avoid overfitting by pruning the tree.
- ML tools: C4.5, C5.0, Weka.

Decision Tree Example



Using Relevance Feedback (Rocchio)

- Relevance feedback methods can be adapted for text categorization.
- Use standard TF/IDF weighted vectors to represent text documents (normalized by maximum term frequency).
- For each category, compute a *prototype* vector by summing the vectors of the training documents in the category.
- Assign test documents to the category with the closest prototype vector based on cosine similarity.

Rocchio Text Categorization Algorithm (Training)

Assume the set of categories is $\{c_1, c_2, \ldots c_n\}$

For i from 1 to n let $\mathbf{p}_i = \langle 0, 0, \ldots, 0 \rangle$

(*initialize prototype vectors*)

For each training example $\langle x, c(x) \rangle \in D$

Let $\mathbf{d}$ be the frequency normalized TF/IDF term vector for doc x

For all i: $(c_i = c(x))$

(*sum all the document vectors in class c_i to get $\mathbf{p}_i$*)

Let $\mathbf{p}_i = \mathbf{p}_i + \mathbf{d}$

Rocchio Text Categorization Algorithm (Test)

Given test document x

Let $\mathbf{d}$ be the TF/IDF weighted term vector for x

Let m = -2 (*init. minimum cosSim*)

For i from 1 to n:

(*compute similarity to each prototype vector*)

Let $s = \text{cosSim}(\mathbf{d}, \mathbf{p}_i)$

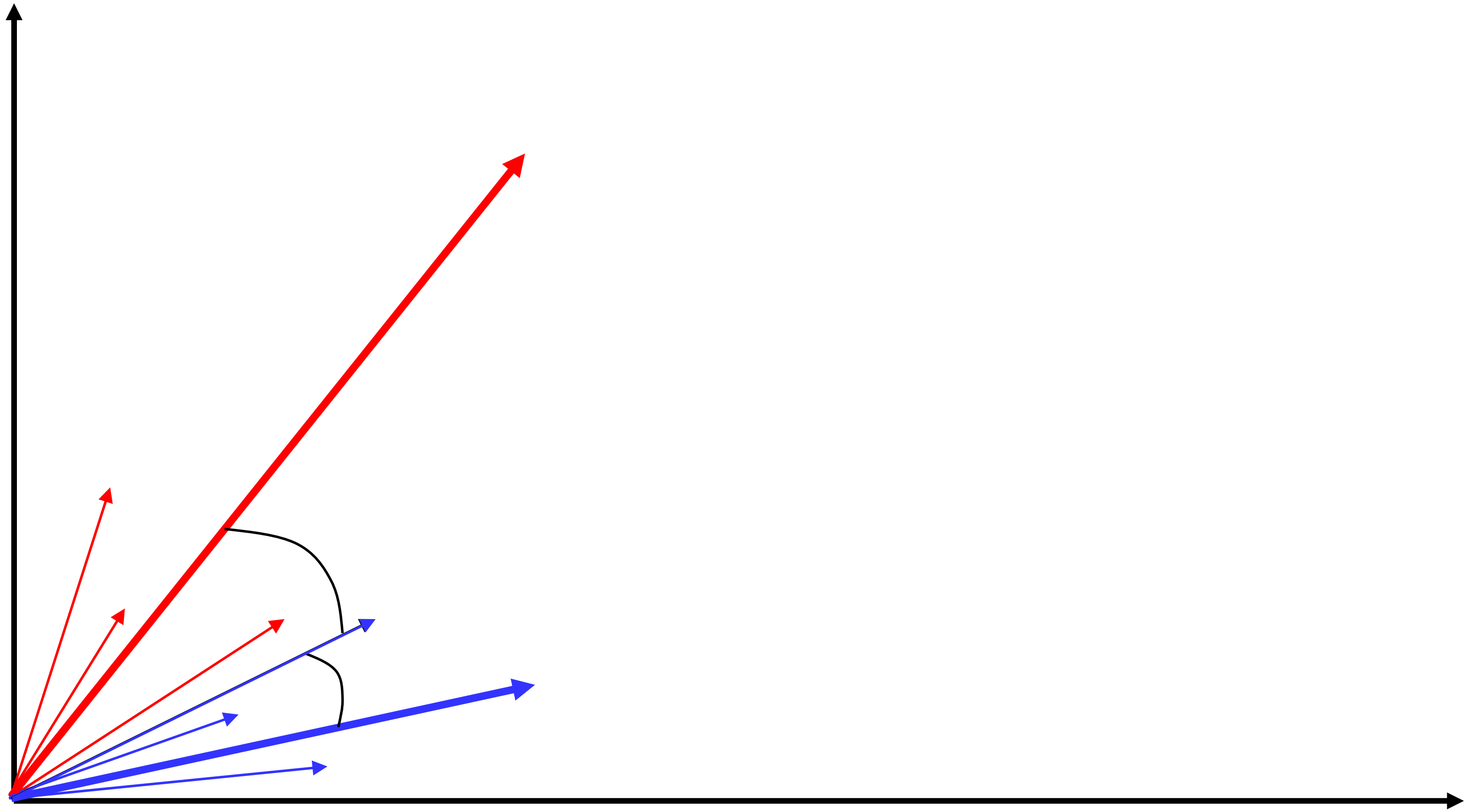
if $s > m$

let $m = s$

let $r = c_i$ (*update most similar class prototype*)

Return class r

Illustration of Rocchio Text Categorization



Exercise 1 (exam preparation :-)

Consider the problem of classifying a name as being Food or Beverage.

Assume the following training set:

- Food: “turkey stuffing”
- Food: “buffalo wings”
- Beverage: “cream soda”
- Beverage: “orange soda”

Apply the Rocchio algorithm to classify a new name:

- “turkey soda”

Rocchio Properties

- Does not guarantee a consistent hypothesis.
- Forms a simple generalization of the examples in each class (a *prototype*).
- Prototype vector does not need to be averaged or otherwise normalized for length since cosine similarity is insensitive to vector length.
- Classification is based on similarity to class prototypes.

Nearest-Neighbor Learning Algorithm

- Learning is just storing the representations of the training examples in *D*.
- Testing instance *x*:
  - Compute similarity between *x* and all examples in *D*.
  - Assign *x* the category of the most similar example in *D*.
- Does not explicitly compute a generalization or category prototypes.
- Also called:
  - Case-based
  - Memory-based
  - Lazy learning

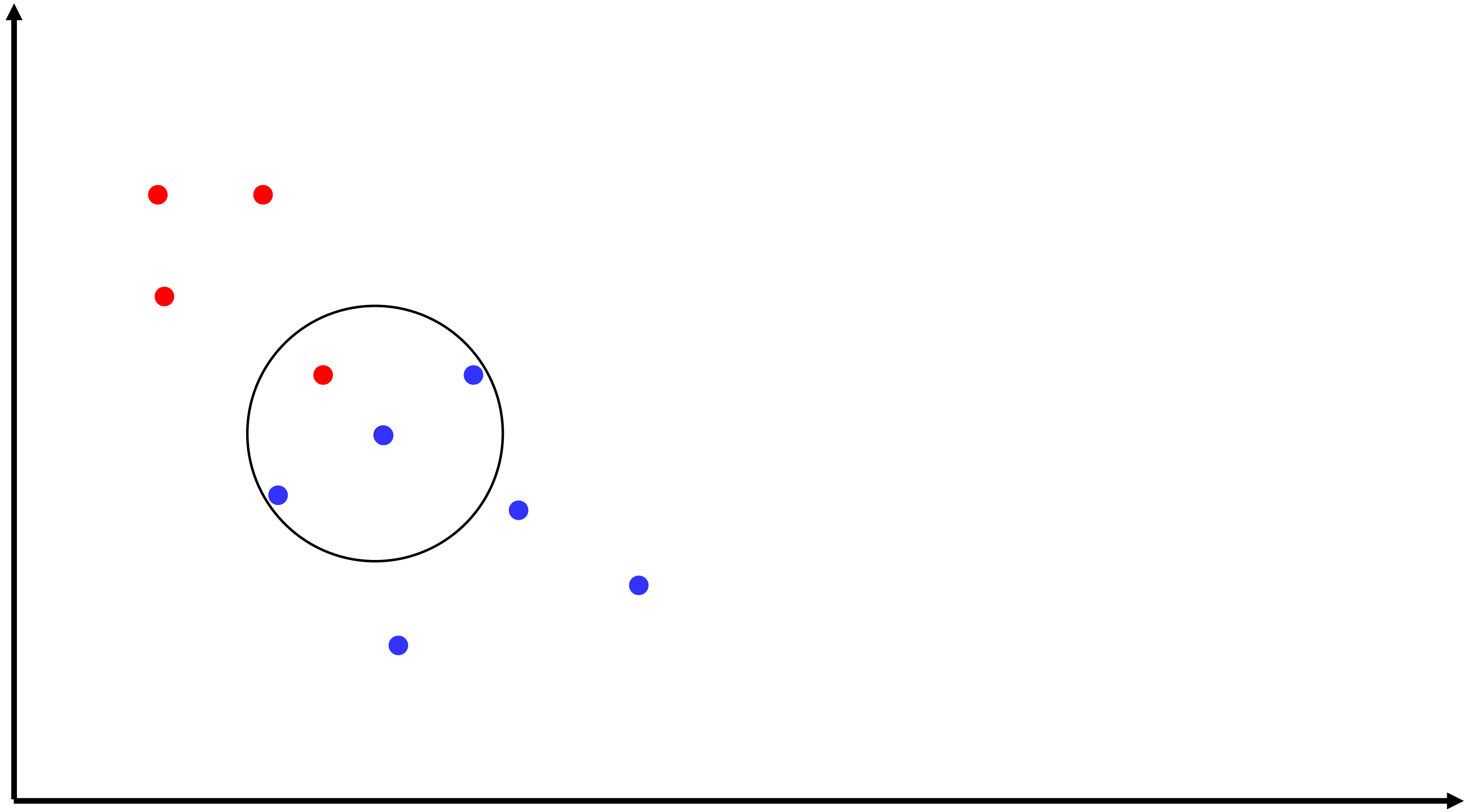
K Nearest-Neighbor

- Using only the closest example to determine categorization is subject to errors due to:
  - A single atypical example.
  - Noise (i.e. error) in the category label of a single training example.
- More robust alternative is to find the *k* most-similar examples and return the majority category of these *k* examples.
- Value of *k* is typically odd to avoid ties, 3 and 5 are most common.

Similarity Metrics

- Nearest neighbor method depends on a similarity (or distance) metric.
- Simplest for continuous *m*-dimensional instance space is *Euclidian distance*.
- For text, cosine similarity of TF-IDF weighted vectors is typically most effective.

3 Nearest Neighbor Illustration
(Euclidian Distance)



K Nearest Neighbor for Text

Training:

For each each training example $<x, c(x)> \in D$

Compute the corresponding TF-IDF vector, $\mathbf{d}_x$, for document x

Test instance *y*:

Compute TF-IDF vector $\mathbf{d}$ for document y

For each $<x, c(x)> \in D$

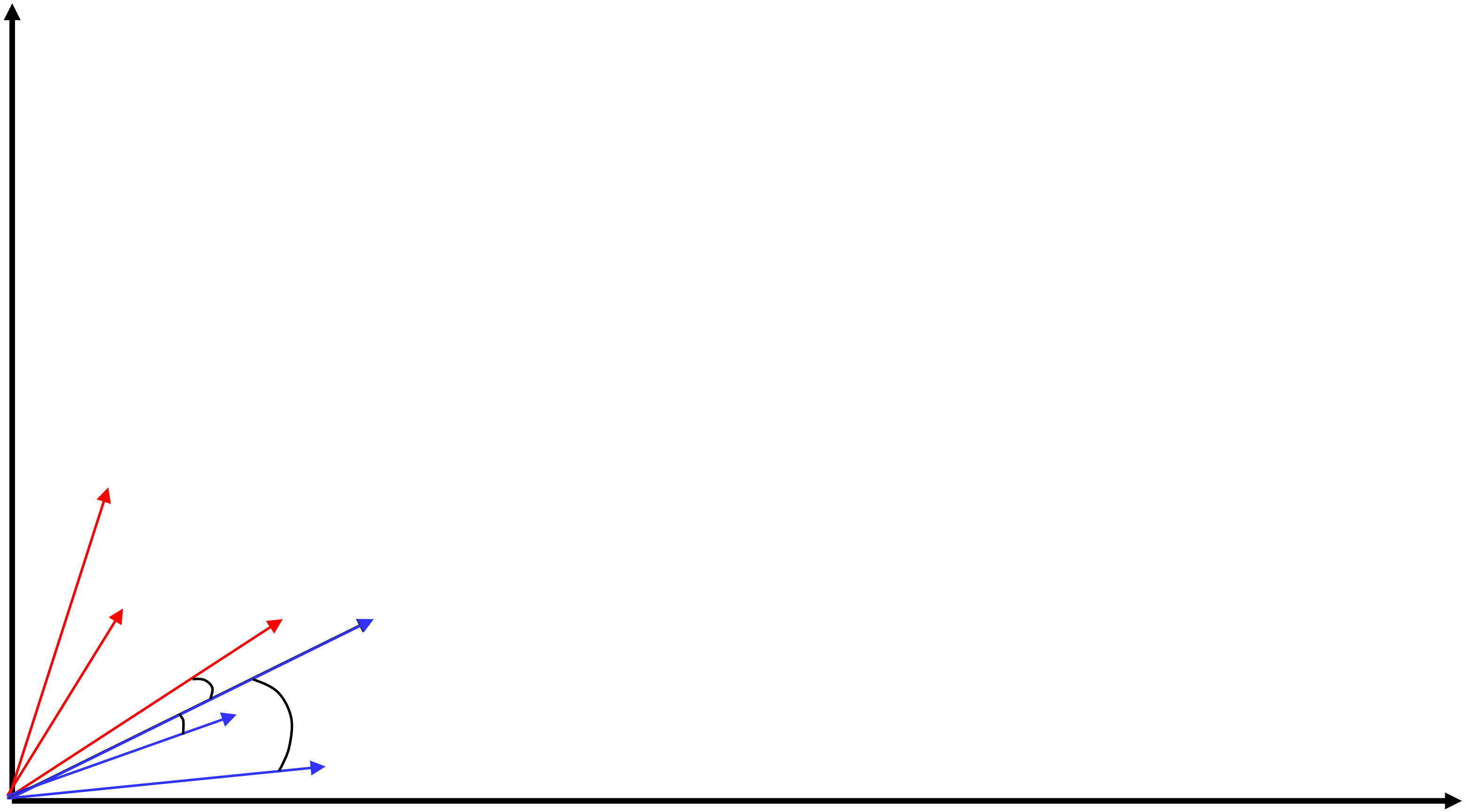
Let $s_x = \text{cosSim}(\mathbf{d}, \mathbf{d}_x)$

Sort examples, x, in D by decreasing value of s_x

Let N be the first k examples in D. (*get most similar neighbors*)

Return the majority class of examples in N

Illustration of 3 Nearest Neighbor for Text



Nearest Neighbor with Inverted Index

- Determining *k* nearest neighbors is the same as determining the *k* best retrievals using the test document as a query to a database of training documents.
- Use standard VSR inverted index methods to find the *k* nearest neighbors.

Exercise 2 (exam preparation :-)

Assume the following training set (2 classes):

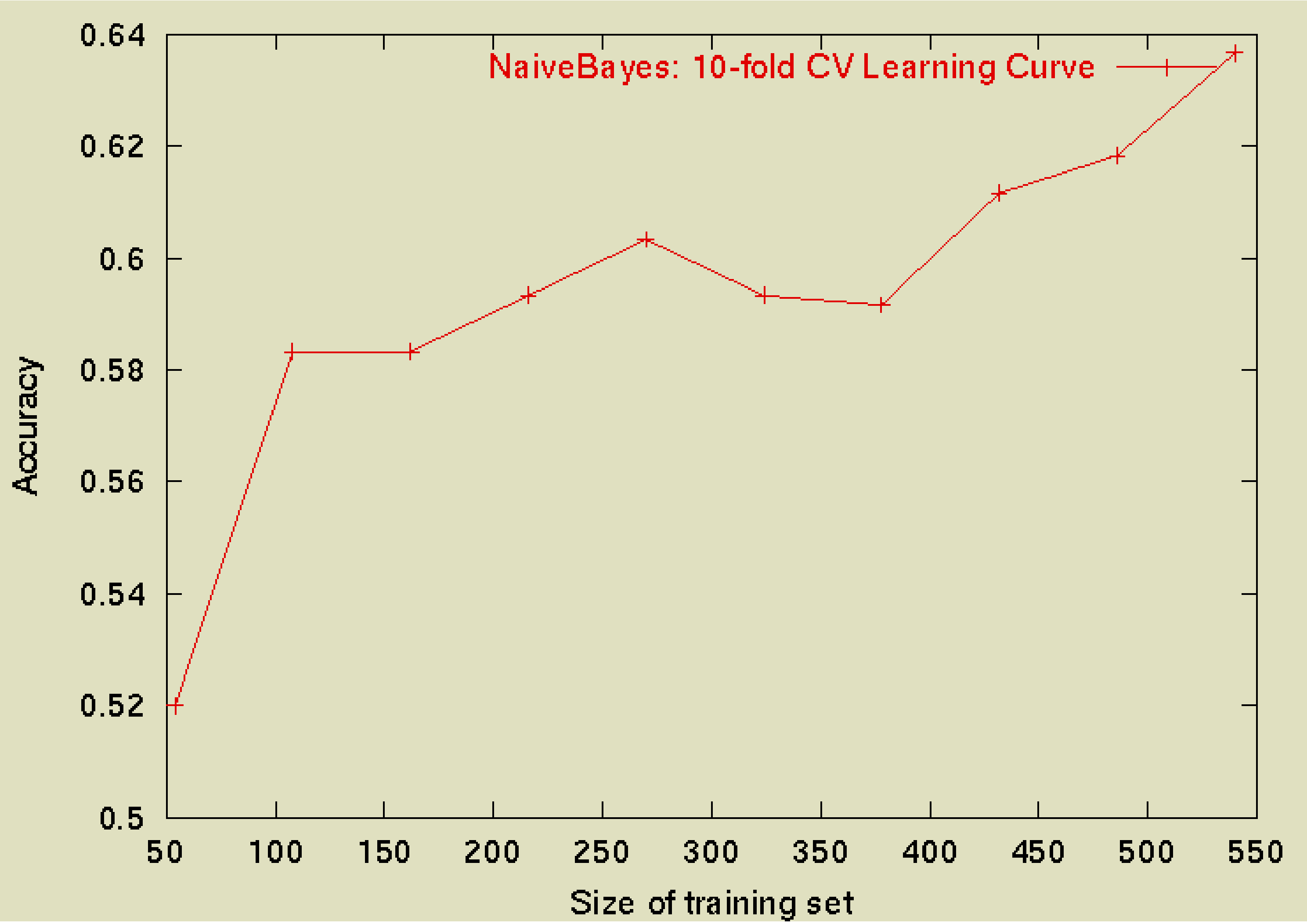
- Food: “turkey stuffing”
- Food: “buffalo wings”
- Beverage: “cream soda”
- Beverage: “orange soda”

Apply kNN with k=3 to classify a new name:

- “turkey soda”

Use tf without idf, with cosine similarity. Would the result be the same if k=1? Why?

Evaluation: Sample Learning Curve (Yahoo Science Data)



Evaluating the results of categorization

- Results on training corpus might not be mirrored in the real world.
- Want to avoid overfitting.
- Need separate test data (hold out 20% of corpus).
- Use N-fold cross-validation (N-1 parts for training and 1 for test, repeat for all partitions)
- Separate development and validation test sets.
- Need measure of performance and comparison to baseline.

Measures of performance

- If binary classification of M texts as members or not members of class c

Predicted Actual	c	not c
c	True Positive TP	False Negative FN
not c	False Positive FP	True Negative TN

Measures of performance

- Accuracy = TP + TN / (TP+FP+TN+FN)
- Precision = TP / (TP + FP)
- Recall = TP / (TP + FN)
- F-measure: trade-off between recall and precision:

$$F = \frac{2PR}{P+R} = \frac{2}{\frac{1}{R}+\frac{1}{P}}$$

- What about more than 2 classes?

Baseline performance

- Baseline: The minimum performance level that you're trying to improve on.
- Could be performance of competing system.
- Could be performance of dumb but easy method:
  - Random choice, most-frequent answer, very simple heuristic, ...
- Comparison should be made on the same test data for results to be fully meaningful.