COEN 169

Text Classification

Yi Fang

Department of Computer Engineering

Santa Clara University

Text Classification

- Task
  - Assign predefined categories to text documents, given the existing documents and their categories
- Motivation: reduce the huge cost of manual text categorization
  - Millions of dollars spent for manual categorization in companies, governments, public libraries, hospitals
  - Manual categorization is almost impossible for some large scale application (classification of Web pages)

Text Classification

- Automatic text categorization
  - Automatically assign predefined categories to text documents
- Procedures
  - Training: Given a set of categories and labeled document examples, learn a method to map a document to correct category
  - Testing: Predict the category of a new document
- Automatic or semi-automatic categorization can significantly reduce the manual efforts

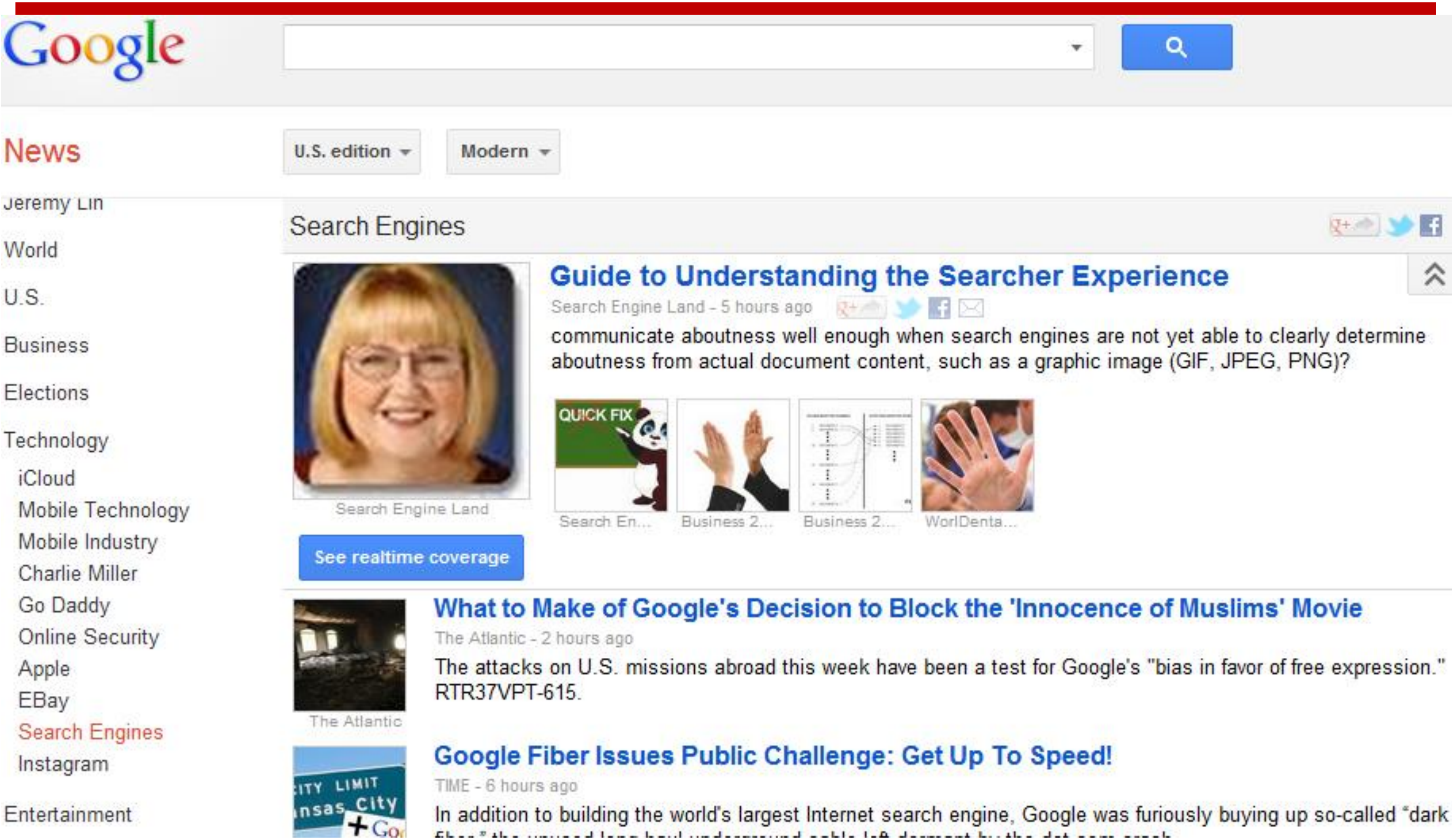
Example: U.S. Census in 1990

- Included 22 million responses
- Needed to be classified into industry categories (200+) and occupation categories (500+)
- Would cost $15 millions if conduced by hand
- Two alternative automatic text categorization methods have been evaluated
  - Knowledge-Engineering (Expert System)
  - Machine Learning (K nearest neighbor method)

Example: U.S. Census in 1990

- A Knowledge-Engineering Approach
  - Expert System (Designed by domain expert)
  - Hand-Coded rules (e.g., if “Professor” and “Lecturer” -> “Education”)
  - Development cost: 2 experts, 8 years (192 Person-months)
  - Accuracy = 47%
- A Machine Learning Approach
  - K Nearest Neighbor (KNN) classification: details later
  - Fully automatic
  - Development cost: 4 Person-months
  - Accuracy = 60%

Google news categorization



Yahoo Directory

YAHOO! DIRECTORY

Web | Directory | Category

Search

National Basketball Association (NBA)

Email this pa

Directory > Recreation > Sports > Basketball > Leagues > **National Basketball Association (NBA)**

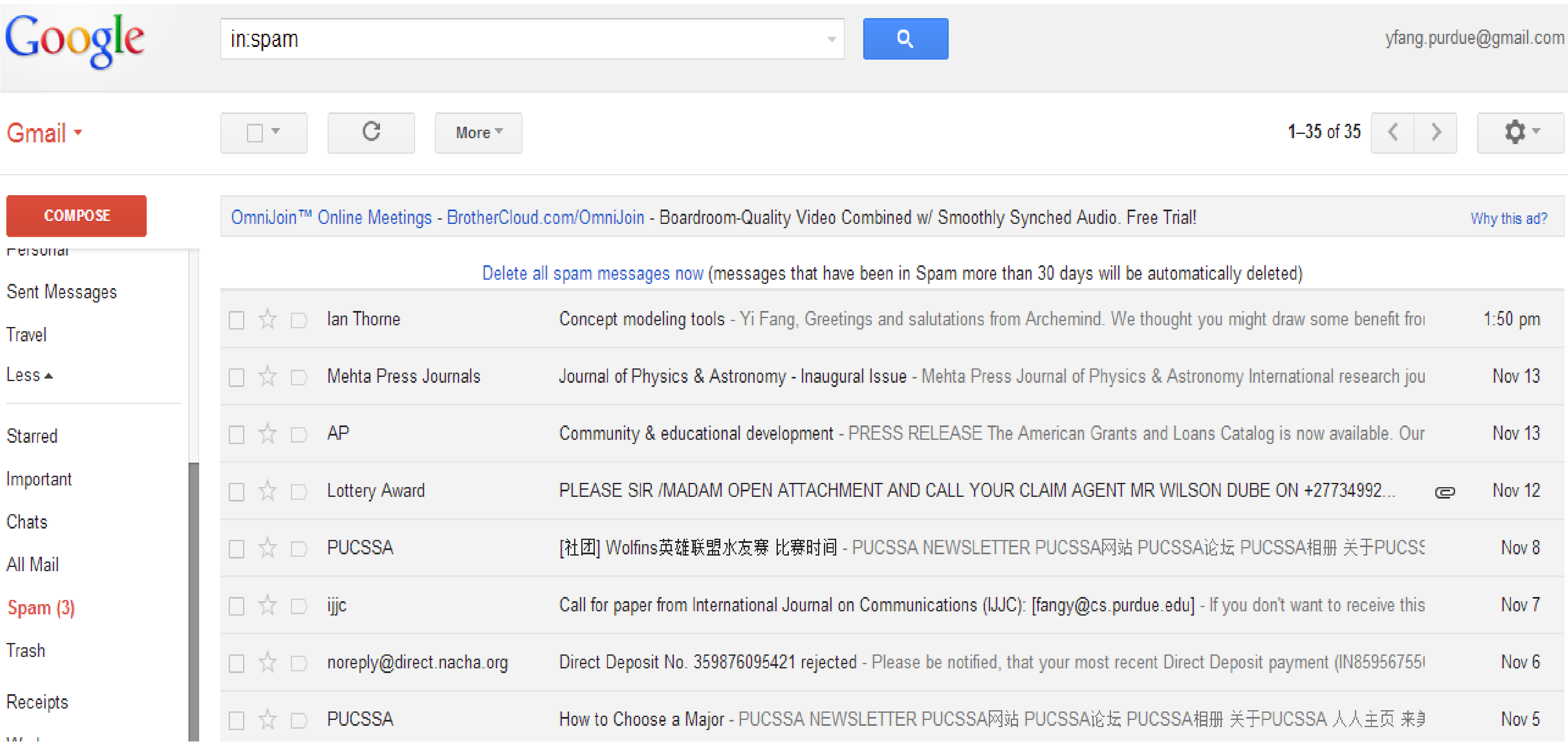
CATEGORIES (What's This?)

- **All-Star Game** (13)
- **Arenas** (29)
- **Coaches** (69)
- **Conferences** (6)
- **Divisions** (30)
- **Draft** (22)
- **Fan Pages** (3)
- **Fantasy** (11)
- **History** (200)
- **National Basketball Development League (NBDL)@**
- **News and Media** (23)
- **Players** (1172)
- **Playoffs** (45)
- **Schedules** (4)
- **Scores** (5)
- **Standings** (3)
- **Statistics** (3)
- **Summer Pro League** (2)
- **Teams** (664)
- **Web Directories** (2)

SITE LISTINGS **By Popularity** | Alphabetical (What's This?) Sites **1 - 2** of **2**

- NBA.com
  Official site of the National Basketball Association. Find current NBA news, scores, schedules, player updates, video highlights, and NBA Finals, Draft, and Draft Lottery coverage.
  www.nba.com

Spam detection



Recall: Vector Space Representation

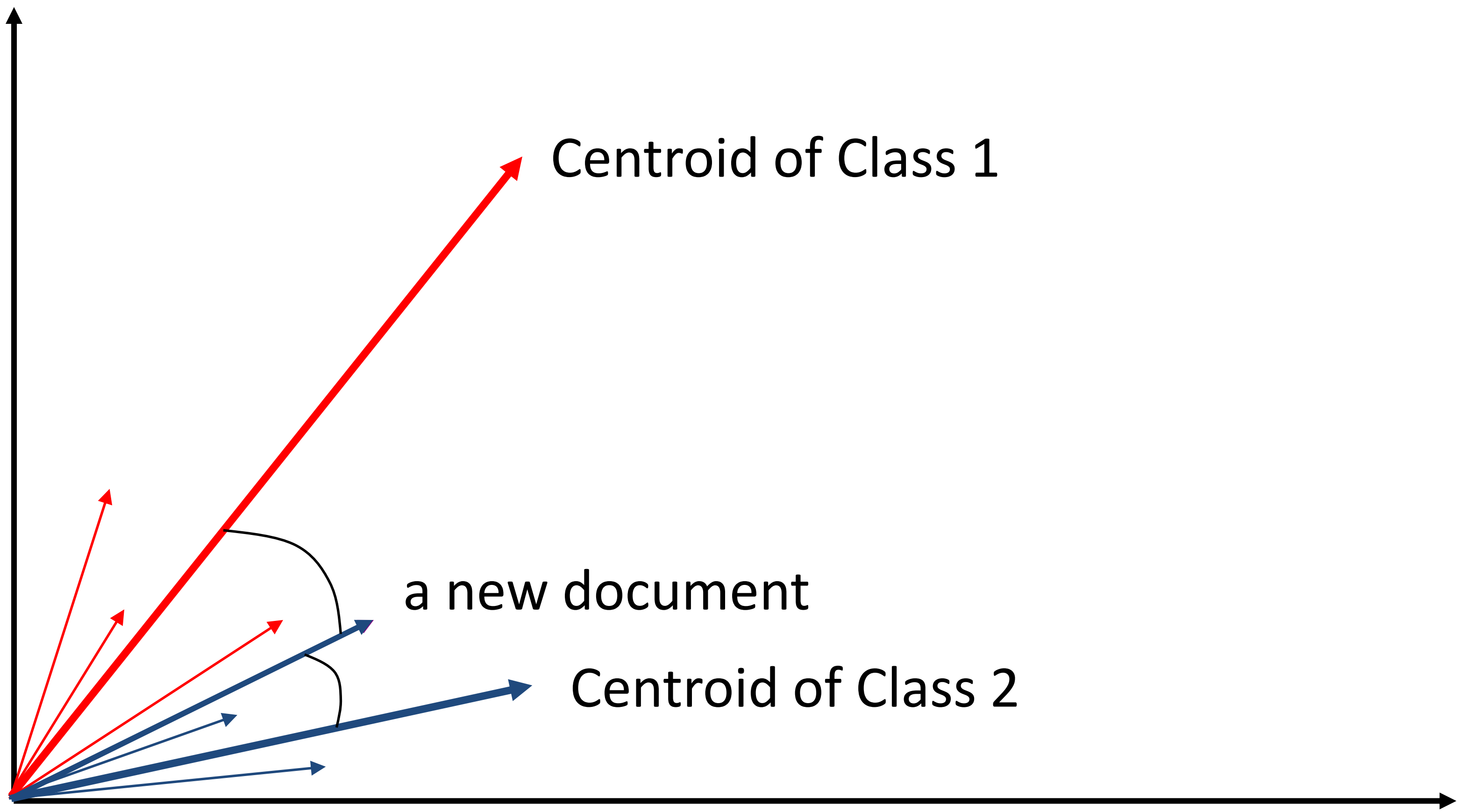
- Each document is a vector, one component for each term (= word).
- High-dimensional vector space:
  - Terms are axes
  - 10,000+ dimensions, or even 100,000+
  - Docs are vectors in this space

- How can we do classification in this space?

Using Rocchio for text classification

- Use standard tf-idf weighted vectors to represent text documents
- For training documents in each category, compute a centroid vector by averaging the vectors of the training documents in the category.
- Assign test documents to the category with the closest centroid vector based on cosine similarity
- Shares similarity with the Rocchio algorithm for relevance feedback introduced in the previous lecture

Illustration of Rocchio Text Categorization



Definition of centroid

$$\vec{\mu}(c) = \frac{1}{|D_c|} \sum_{d \in D_c} \vec{v}(d)$$

- Where D_c is the set of all documents that belong to class *c* and *v*(*d*) is the vector space representation of *d*.

Rocchio classification

- Forms a simple representation for each class: the centroid
- The assumption is violated if there exist polymorphic categories
- It is little used outside text classification
  - It has been used quite effectively for text classification
  - But in general worse than the other methods
  - Efficient and easy to implement

K-Nearest Neighbor Classifier

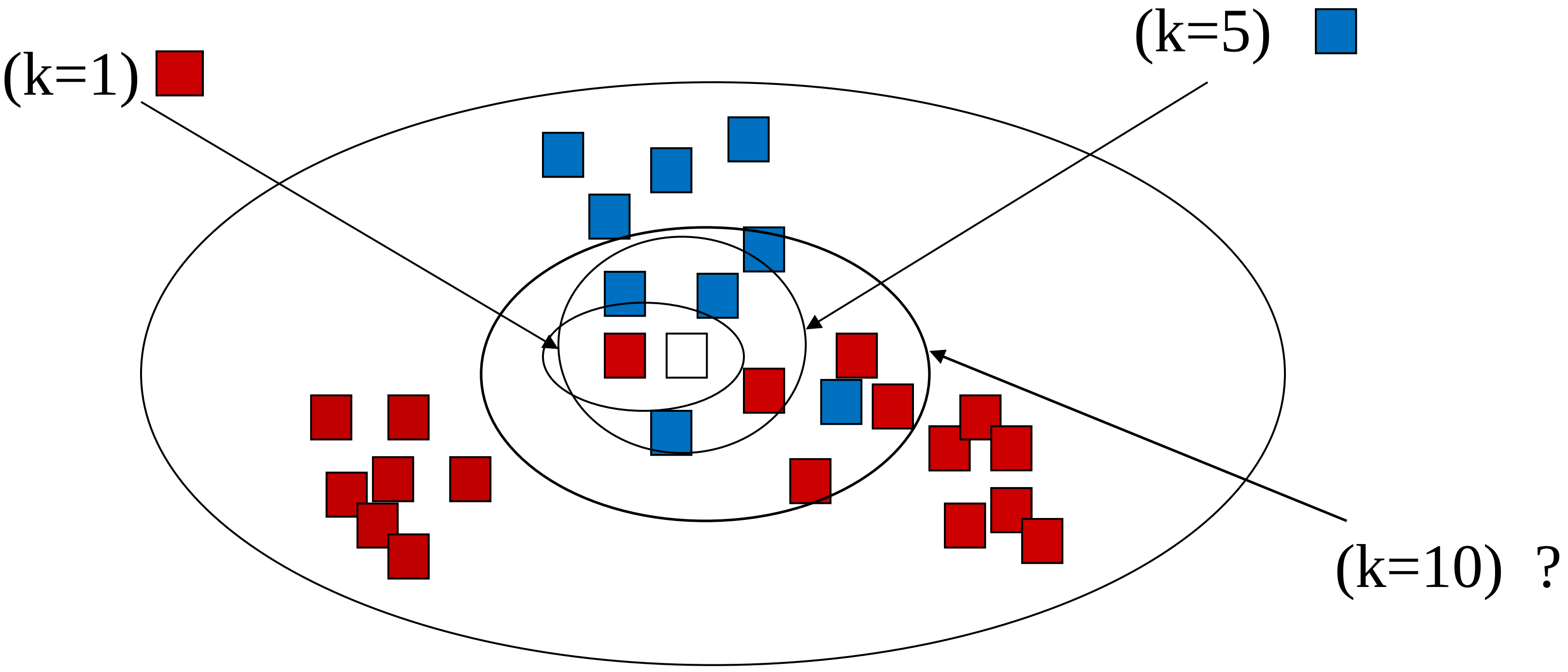
- Commonly used in data mining
- low/no cost in “training”, high cost in prediction
- Among top-performing text categorization methods

K-Nearest Neighbor Classifier

1. Keep all training documents
2. Find *k* documents that are most similar to the new document (“neighbor” documents)
3. Assign the category that is most common in these neighbor documents (neighbors vote for the category)

K-Nearest Neighbor Classifier

Idea: find your label by what label your neighbors use



- Use K nearest neighbors to vote

1-NN: Red; 5-NN: Blue; 10-NN: ?; Weight 10-NN: Blue

K Nearest Neighbor: Framework

Considering the distance of a neighbor (A closer neighbor has more weight/influence)

Training data $D = \{(x_i, y_i)\}, \quad x_i \in R^M, \text{docs}, \quad y_i \in \{0,1\}$

Test data $x \in R^M$ The neighborhood is D_k

Scoring Function

$$\hat{y}(x) = \frac{1}{\sum_{x_i \in D_k(x)} sim(x, x_i)} \sum_{x_i \in D_k(x)} sim(x, x_i) y_i$$

Classification:

$$\begin{cases} 1, & \text{if } \hat{y}(x) > 0.5 \\ 0, & \text{otherwise} \end{cases}$$

Document Representation: X_i uses tf.idf weighting for each dimension

K Nearest Neighbor: Technical Elements

- Document representation
- Document distance measure: closer documents should have similar labels
- Number of nearest neighbors (value of K)

Choices of Similarity Functions

Euclidean distance $d(\vec{x}_1, \vec{x}_2) = \sqrt{\sum_v (x_{1v} - x_{2v})^2}$

KL divergence $d(\vec{x}_1, \vec{x}_2) = \sum_v x_{1v} \log \frac{x_{1v}}{x_{2v}}$

Dot product $\vec{x}_1 * \vec{x}_2 = \sum_v x_{1v} * x_{2v}$

Cosine Similarity $\cos(\vec{x}_1, \vec{x}_2) = \frac{\sum_v x_{1v} * x_{2v}}{\sqrt{\sum_v x_{1v}^2}\sqrt{\sum_v x_{2v}^2}}$

For text classification, cosine similarity of tf-idf weighted vectors is typically most effective

Choices of Number of Neighbors (K)

Find desired number of neighbors by cross validation

- Choose a subset of available data as training data, the rest as validation data
- Find the desired number of neighbors on the validation data
- The procedure can be repeated for different splits; find the consistent good number for the splits

Characteristics of KNN

Pros

- Simple and intuitive,
- Widely used and provide strong baseline in TC Evaluation
- Easy to implement; can use standard IR techniques (e.g., tf-idf)

Cons

- Heuristic approach, no explicit objective function
- Difficult to determine the number of neighbors
- High online cost in testing; find nearest neighbors has high time complexity

Naïve Bayes Text Classification

- Essentially the statistical language modeling approach that we have learned in the previous lecture
- Concatenate all the documents of a category into a “big document”
- Treat the new document as a query
- Compute the query likelihood
- Consider the class prior

Bayes' Rule

Use *C* represents a class and *d* represents a document

$$P(C,d) = P(C \mid d)P(d) = P(d \mid C)P(C)$$

$$\boxed{P(C \mid d) = \frac{P(d \mid C)P(C)}{P(d)}}$$

- Treat *C* as the big document that combines all the documents in the class
- Build the language model for *C*
- Treat *d* as the query
- We can then compute $P(d \mid C)$ by the query likelihood method in the previous lecture
- $P(C)$ is class prior

Naive Bayes Classifiers

$$C = \underset{C_j}{\operatorname{argmax}} P(C_j \mid d)$$

$$= \underset{C_j}{\arg\max} \frac{P(d \mid C_j) P(C_j)}{P(d)}$$

$$= \underset{C_j}{\operatorname{argmax}} \frac{P(t_1, t_2, \ldots, t_n \mid c_j) P(C_j)}{P(t_1, t_2, \ldots, t_n)}$$

$$= \underset{C_j}{\operatorname{argmax}} P(t_1, t_2, \ldots, t_n \mid C_j) P(C_j)$$

$$= \underset{C_j}{\operatorname{argmax}} P(C_j) \prod_{i=1}^{n} P(t_i \mid C_j)$$

Learning the Model

- First attempt: maximum likelihood estimates
  - simply use the frequencies in the data

$$P(t_i \mid C_j) = \frac{tf(t_i, C_j)}{|C_j|}$$

$$P(C_j) = \frac{\#of\ \ docs\ in\ C_j}{total\#of\ \ docs}$$

Smoothing

$$P(t_i \mid C_j) = \frac{tf(t_i, C_j) + 1}{|C_j| + |V|}$$

- This is just add-one smoothing!
- You can alternatively throw in any of the other smoothing techniques we have learned in statistical language modeling

Naïve Bayes

- From training corpus, extract *Vocabulary*
- Calculate required $P(C_j)$ and $P(t_i | C_j)$ terms
  - For each C_j in C do
    - $docs_j \leftarrow$ subset of documents for which the target class is C_j
    - $$P(C_j) \leftarrow \frac{|docs_j|}{|\text{total \# documents}|}$$

  $Text_j \leftarrow$ single big document containing all $docs_j$
  for each word t_i in *Vocabulary*

  $n_i \leftarrow$ number of occurrences of t_i in $Text_j$

  $$P(t_i | C_j) \leftarrow \frac{n_i + 1}{|C_j| + |Vocabulary|}$$

Evaluating classification

- Evaluation must be done on test data that are independent of the training data (a disjoint set of instances)

- Measures: Precision, recall, *F*, classification accuracy

Evaluation metrics

	in the class	not in the class
predicted to be in the class	true positives (TP)	false positives (FP)
predicted to not be in the class	false negatives (FN)	true negatives (TN)

Precision = TP / (TP + FP)

Recall = TP / (TP + FN)

F = 2 Precision * Recall/(Precision + Recall)*

Accuracy = (TP + TN)/(TP+FP+FN+TN)

Evaluation metrics

- Example: classify documents into spam or not spam

	system's prediction	correct answer	TP	FP	FN	TN
d1	→ Y	N		1		
d2	→ Y	Y	1			
d3	→ N	Y			1	
d4	→ N	N				1
d5	→ Y	N		1		

Evaluation metrics

- Example: classify documents into spam or not spam

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} = \frac{1}{1+2} = 0.333$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} = \frac{1}{1+1} = 0.5$$

$$F = \frac{2 \cdot \text{Recall} \cdot \text{Precision}}{\text{Recall+Precision}} = \frac{2 \cdot 1/3 \cdot 1/2}{1/3 + 1/2} = 0.4$$

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+FP+FN+TN}} = \frac{1+1}{1+2+1+1} = 0.4$$