Introduction to

Information Retrieval

Supervised Methods
in Information Retrieval

Categorization/Classification

- Given:
  - A representation of a document *d*
    - Issue: how to represent text documents.
    - Usually some type of high-dimensional space – bag of words, vector space
  - A fixed set of classes:

    $C = \{c_1, c_2, \ldots, c_J\}$
- Determine:
  - The category of *d*: $\gamma(d) \in C$, where $\gamma(d)$ is a classification function
  - We want to build classification functions ("classifiers").

Classification examples

- Classify an email as spam / legitimate
- Classify a news article as per its topic (Politics / Sports / Entertainment / ..)

Classification Methods (1)

- Manual classification
  - Used by the original Yahoo! Directory
  - Accurate when job is done by experts
  - Consistent when the problem size and team is small
  - Difficult and expensive to scale
    - Means we need automatic classification methods for big problems

Classification Methods (2)

- Hand-coded rule-based classifiers
  - One technique used by news agencies, intelligence agencies, government, etc.
  - Accuracy can be high if a rule has been carefully refined over time by a subject expert
  - Building and maintaining these rules is expensive

Classification Methods (3): Supervised learning

- Given:
  - A document *d*
  - A fixed set of classes:

    $C = \{c_1, c_2, \ldots, c_J\}$
  - A training set *D* of documents each with a label in *C*
- Determine:
  - A learning method or algorithm which will enable us to learn a classifier γ
  - For a test document *d*, we assign it the class

$\gamma(d) \in C$

Classification Methods (3)

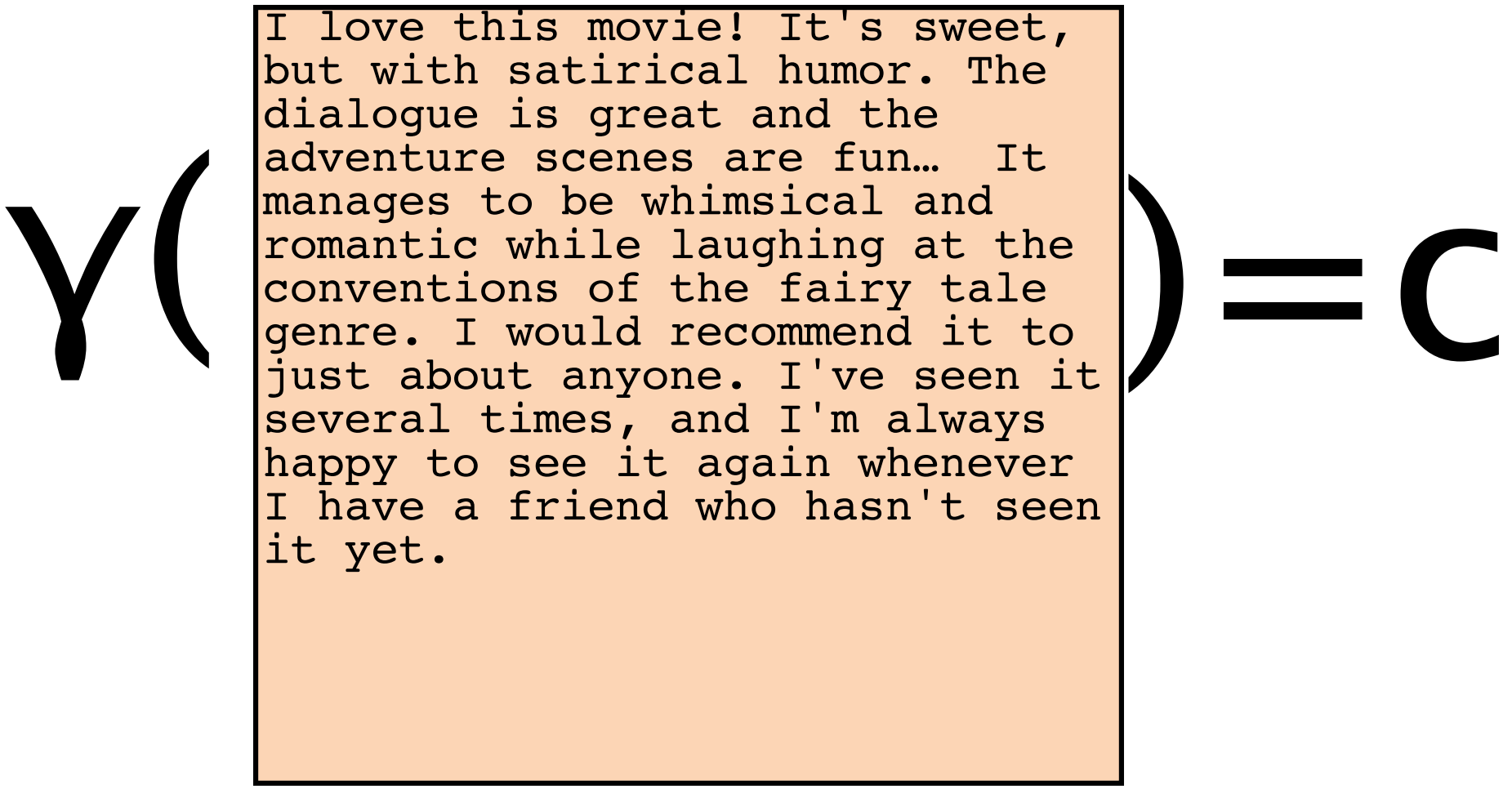
- Supervised learning
  - Naive Bayes (simple, common)
  - k-Nearest Neighbors (simple, powerful)
  - Support-vector machines (generally more powerful)
  - Decision trees → random forests → gradient-boosted decision trees (e.g., xgboost)
  - Neural models
  - … plus many other methods

- No free lunch: need hand-classified training data
- Many commercial systems use a mix of methods

How to classify? Based on Features

- Supervised learning classifiers can use any sort of feature
  - URL, email address, punctuation, capitalization, dictionaries, network features
- In the simplest bag of words view of documents
  - We use **only** word features

The bag of words representation



The bag of words representation

Remember: Vector Space Representation

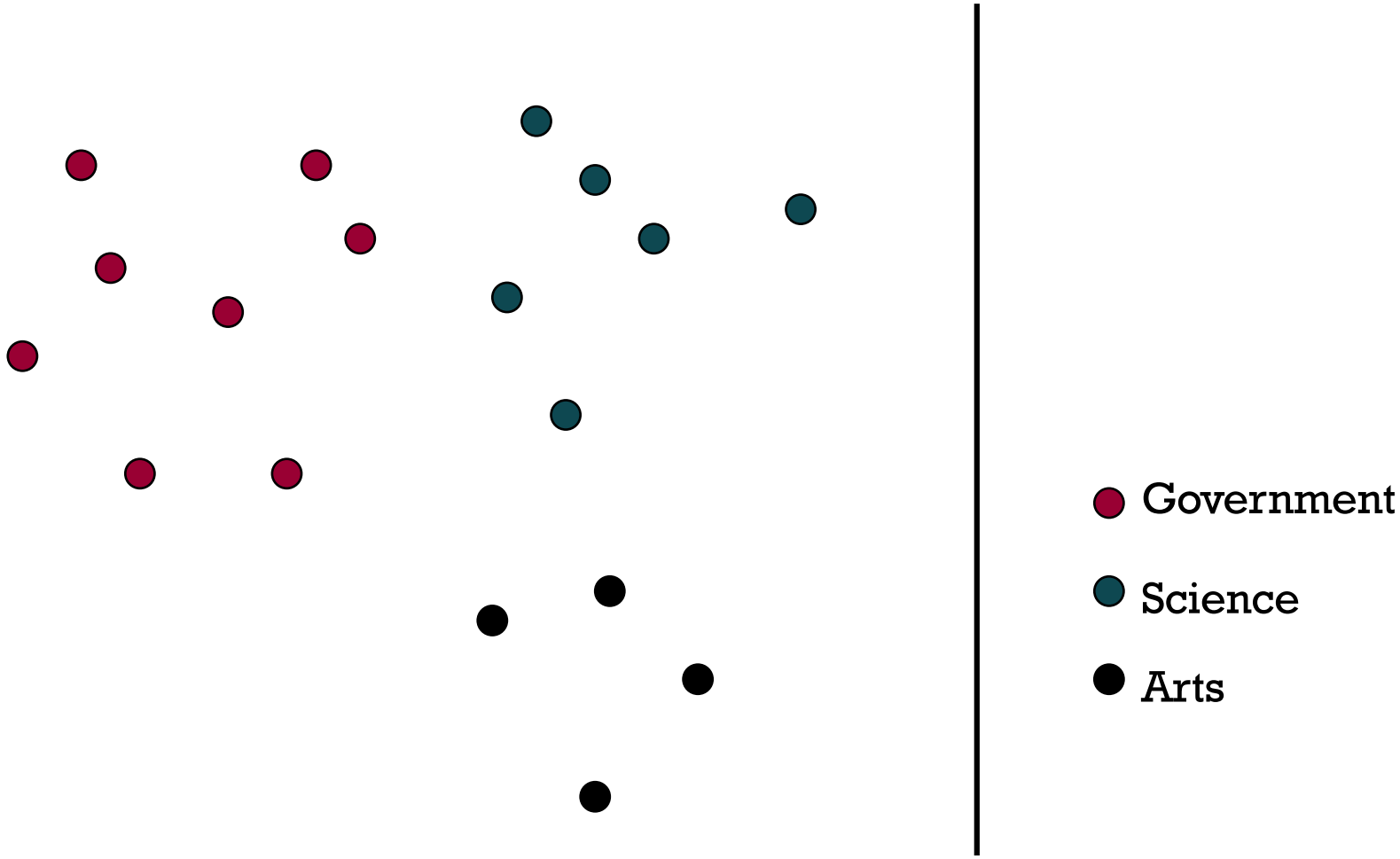
- Each document is a vector, one component for each term (= word).
- Normally normalize vectors to unit length.
- High-dimensional vector space:
  - Terms are axes
  - 10,000+ dimensions, or even 100,000+
  - Docs are vectors in this space

- How can we do classification in this space?

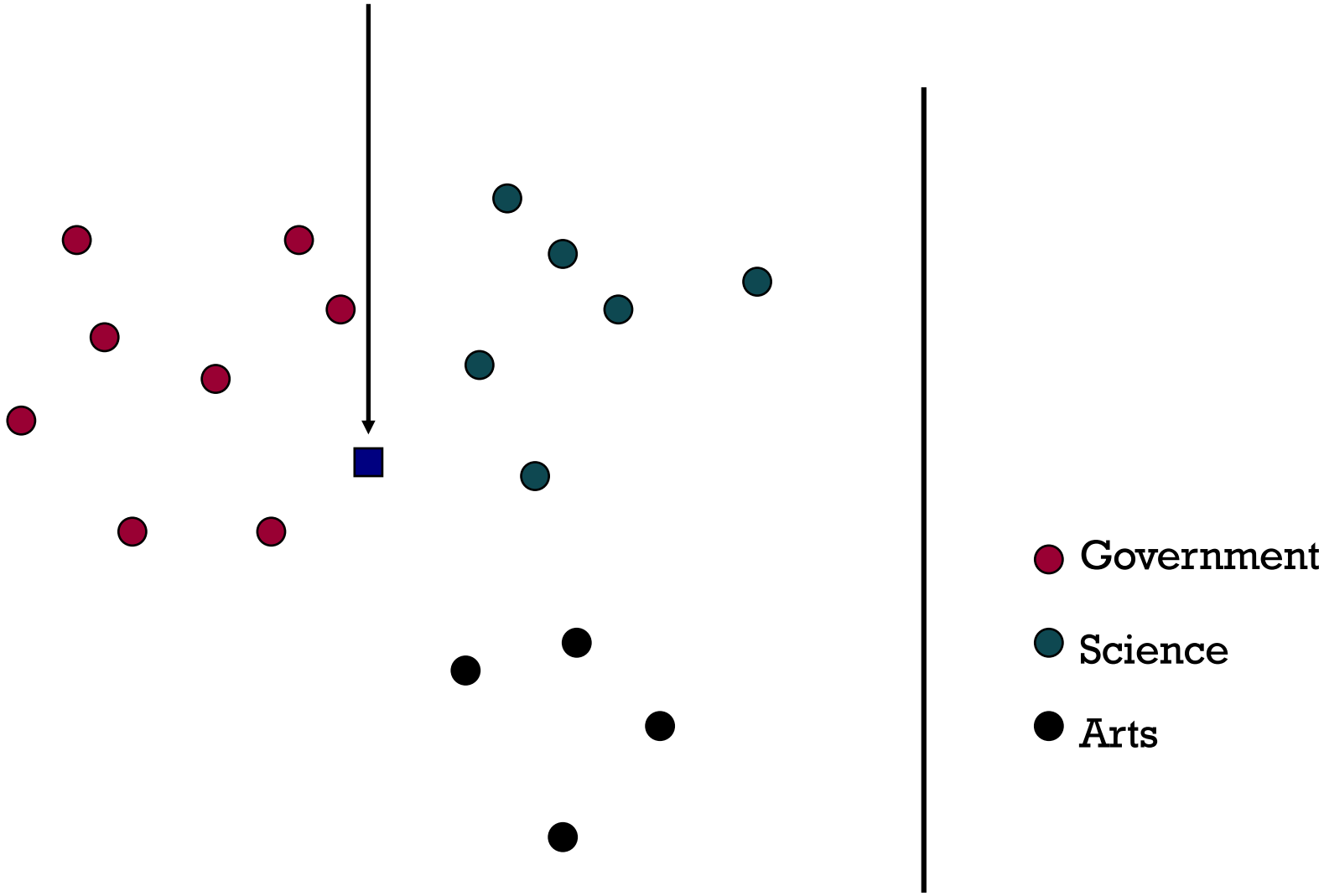
Classification Using Vector Spaces

- In vector space classification, training set corresponds to a labeled set of points (equivalently, vectors)
- Premise 1: Documents in the same class form a contiguous region of space
- Premise 2: Documents from different classes don't overlap (much)
- Learning a classifier: build surfaces to delineate classes in the space

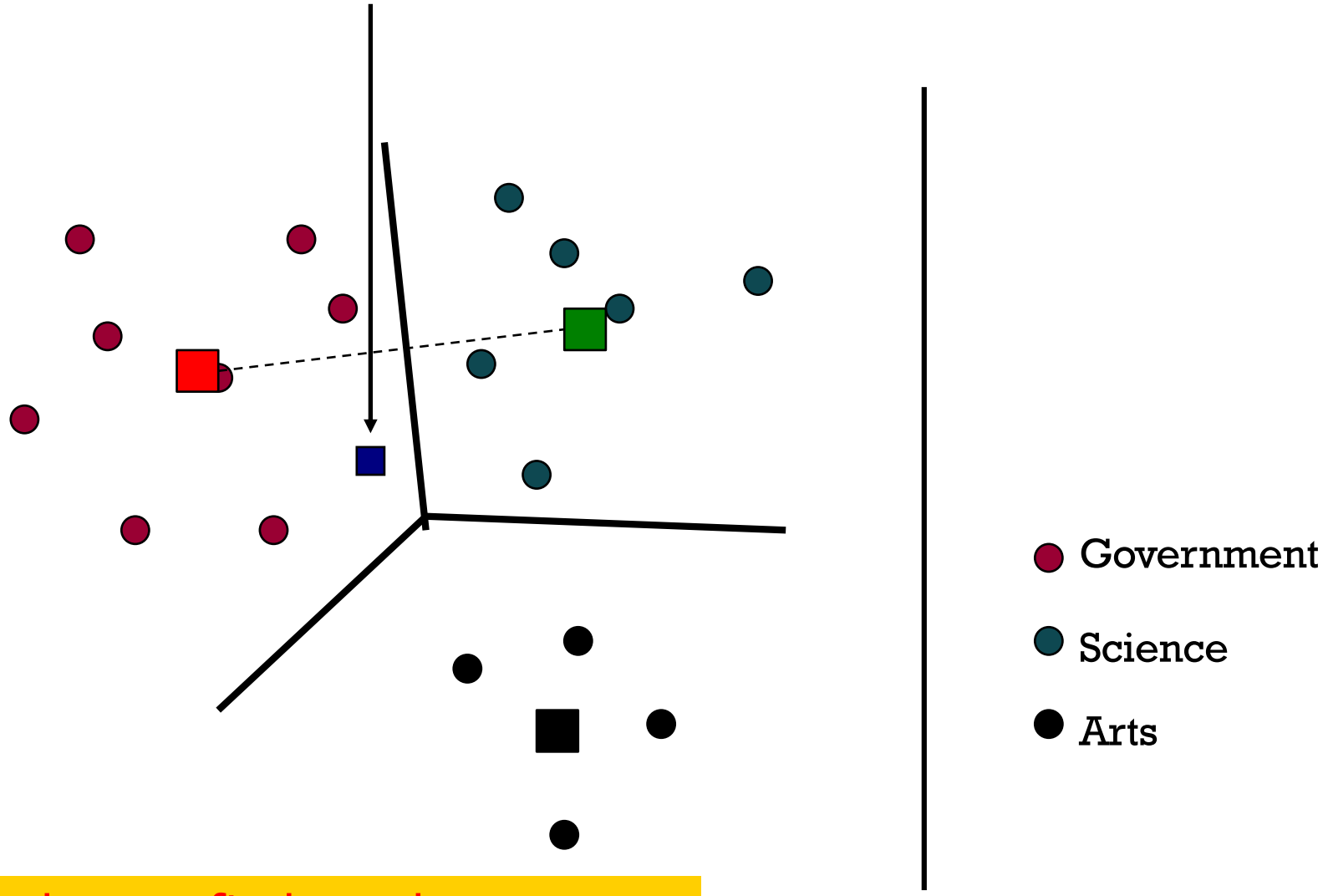
Documents in a Vector Space



Test Document of what class?



Test Document = Government



Our focus: how to find good separators

Another supervised learning problem: Regression

- Instead of mapping instances (e.g., documents) to discrete labels/classes …
- Map instances to a continuous (real) value

- E.g.,
  - Given the features (e.g., height, food habit, occupation, weight) of many individuals, predict the weight of a new person for whom the other features are known
  - Given the bag of words representation of an email, predict the probability of it being spam

SUPERVISED METHODS FOR RETRIEVAL

Machine learning for IR ranking?

- We've looked at methods for ranking documents in IR
  - Cosine similarity, inverse document frequency, BM25, proximity, pivoted document length normalization, (will look at) Pagerank, …
- We've discussed supervised learning problems – classification and regression

- Can we can use *machine learning* to rank the documents displayed in search results?
  - Sounds like a good idea
  - Known as "machine-learned relevance" or "learning to rank"
  - Actively researched and used by Web search engines

Why wasn't ML much needed earlier?

- Traditional ranking functions in IR used a very small number of features, e.g.,
  - Term frequency
  - Inverse document frequency
  - Document length

- It was ~~easy~~ possible to tune weighting coefficients by hand
  - And people did so

Why is ML needed now?

- Modern (web) systems use a great number of features:
  - Log frequency of query word in anchor text?
  - Query word in color on page?
  - # of images on page?
  - # of (out) links on page?
  - PageRank of page?
  - URL length?
  - URL contains "~"?
  - Page edit recency
  - Page loading speed

- The *New York Times* in 2008-06-03 quoted Amit Singhal as saying Google was using over 200 such features ("signals") – so it's sure to be over 500 today. ☺

Simple example: Using classification for ad hoc IR

- Collect a training corpus of (*q, d, r*) triples
  - Relevance *r* is here binary (but may be multiclass, with 3–7 values)
  - Query-Document pair is represented by a feature vector
  - Train a machine learning model to predict the class *r* of a document-query pair

“Learning to rank”

- Classification probably isn’t the right way to think about approaching ad hoc IR:
  - Classification problems: Map to an unordered set of classes
  - Regression problems: Map to a real value
  - Ordinal regression (or “ranking”) problems: Map to an *ordered* set of classes

“Learning to rank”

- Assume a number of categories ***C*** of relevance exist
  - These are totally ordered: $c_1 < c_2 < \ldots < c_J$
  - This is the ordinal regression setup

- Assume training data is available consisting of document-query pairs (d, q) represented as feature vectors x_i with relevance ranking c_i

Yahoo! Learning to Rank Challenge

- Yahoo! Webscope dataset : 36,251 queries, 883k documents, 700 features, 5 ranking levels
  - Ratings: Perfect, Excellent, Good, Fair, Bad
  - Real web data from U.S. and “an Asian country”
  - set-1: 473,134 feature vectors; 519 features; 19,944 queries
  - set-2: 34,815 feature vectors; 596 features; 1,266 queries

Yahoo! Learning to rank challenge

- Goal was to validate learning to rank methods on a large, “real” web search problem

- Only feature vectors released
  - Not URLs, queries, nor feature descriptions
    - Wanting to keep privacy and proprietary info safe
  - But included web graph features, click features, page freshness and page classification features as well as text match features

SUPERVISED METHODS FOR SUMMARIZATION

Supervised summarization

- Supervised techniques use a collection of documents and human-generated summaries for them to train a classifier for the given text

- Sentences in an original training document can be labelled as “in summary” or “not in summary”

- Features of sentences (e.g. position of the sentence, number of keywords in the sentence, etc.) that make them good candidates for inclusion in the summary are learnt

Supervised summarization

- The main drawback of supervised techniques is that training data is expensive to produce and relatively sparse

- Also, most readily available human-generated summaries are abstractive in nature
  - How to decide the label of sentences in the source document if the reference summary is abstractive?