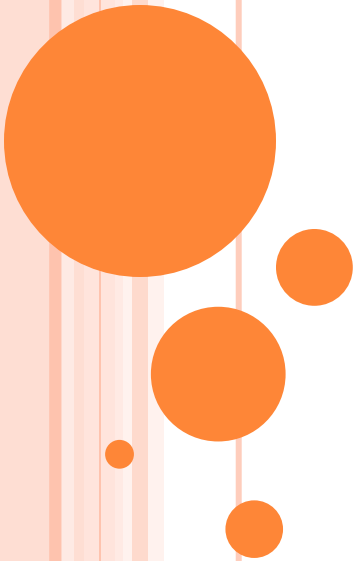


Text Classification

Standing queries

- The path from IR to text classification:
  - You have an information need to monitor, say:
    - Unrest in the Niger delta region
  - You want to rerun an appropriate query periodically to find new news items on this topic
  - You will be sent new documents that are found
    - i.e., it's not ranking but classification (relevant vs. not relevant)
- Such queries are called **standing queries**
  - Long used by "information professionals"
  - A modern mass instantiation is **Google Alerts**
- Standing queries are (hand-written) text classifiers

From: Google Alerts
Subject: **Google Alert - stanford -neuro-linguistic nlp OR "Natural Language Processing" OR parser OR tagger OR ner OR "named entity" OR segmenter OR classifier OR dependencies OR "core nlp" OR corenlp OR phrasal**
Date: May 7, 2012 8:54:53 PM PDT
To: Christopher Manning

Web

3 new results for **stanford -neuro-linguistic nlp OR "Natural Language Processing" OR parser OR tagger OR ner OR "named entity" OR segmenter OR classifier OR dependencies OR "core nlp" OR corenlp OR phrasal**

Twitter / **Stanford NLP** Group: @Robertoross If you only n ...
@Robertoross If you only need tokenization, java -mx2m edu.**stanford.nlp**. process.PTBTokenizer file.txt runs in 2MB on a whole file for me.... 9:41 PM Apr 28th ...
twitter.com/stanfordnlp/status/196459102770171905

[Java] LexicalizedParser lp = LexicalizedParser.loadModel("edu ...
loadModel("edu/**stanford/nlp**/models/lexparser/englishPCFG.ser.gz");. String[] sent = { "This", "is", "an", "easy", "sentence", "." };. Tree parse = lp.apply(Arrays.
pastebin.com/az14R9nd

More Problems with Statistical **NLP** || kuro5hin.org
Tags: nlp, ai, coursera, **stanford**, **nlp**-class, cky, nltk, reinventing the wheel, ... Programming Assignment 6 for **Stanford's nlp**-class is to implement a CKY parser .
www.kuro5hin.org/story/2012/5/5/11011/68221

Tip: Use quotes ("like this") around a set of words in your query to match them exactly. Learn more.

Delete this alert.
Create another alert.
Manage your alerts.

Spam filtering
Another text classification task

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 !

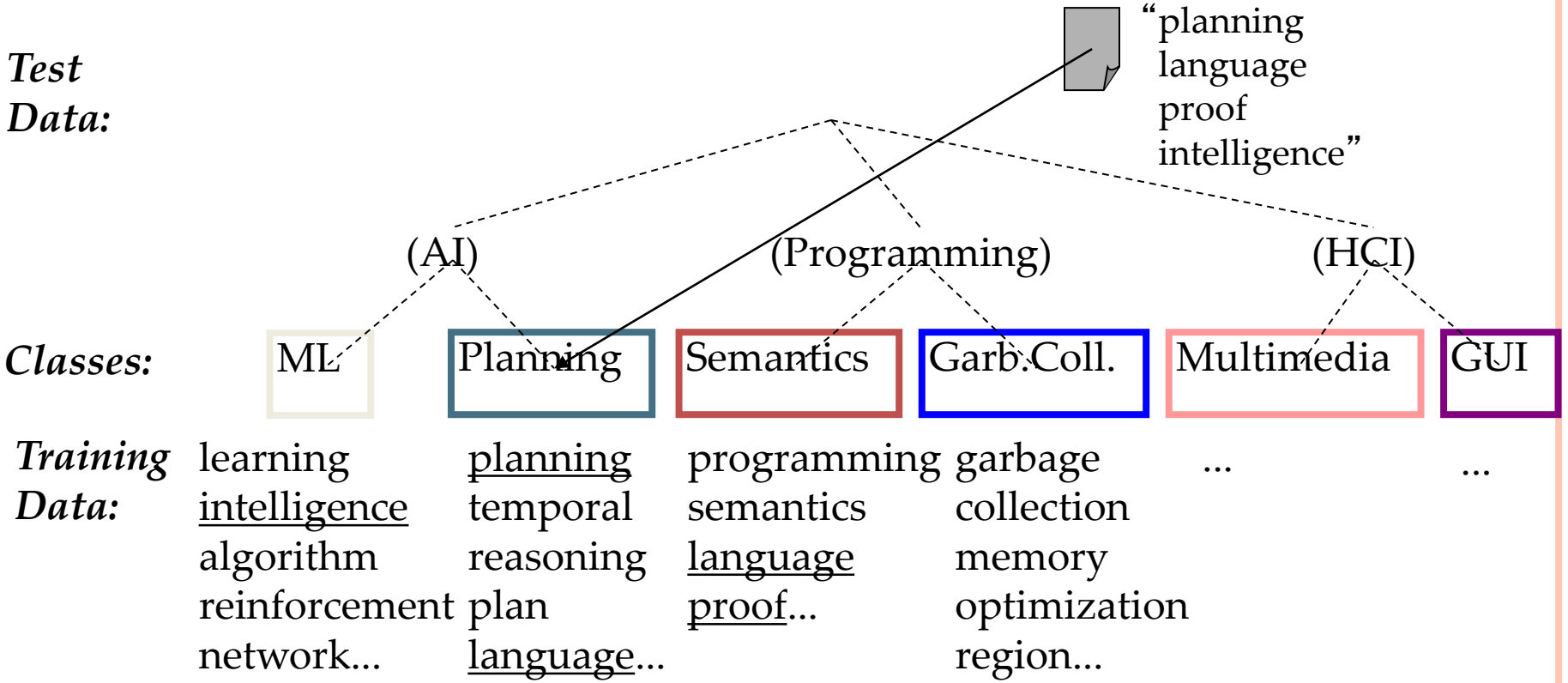
==========
Click Below to order:
http://www.wholesaledaily.com/sales/nmd.htm
==========

Categorization/Classification

- Given:
  - A representation of a document *d*
    - Issue: how to represent text documents.
    - Usually some type of high-dimensional space – bag of words
  - 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").

Document Classification



Classification Methods (1)

- Manual classification
  - Used by the original Yahoo! Directory
  - Looksmart, about.com, ODP, PubMed
  - 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, etc.
  - Widely deployed in government and enterprise
  - Vendors provide "IDE" for writing such rules

Classification Methods (2)

- Hand-coded rule-based classifiers
  - Commercial systems have complex query languages
  - Accuracy can be high if a rule has been carefully refined over time by a subject expert
  - Building and maintaining these rules is expensive

A Verity topic
A complex classification rule

```
comment line          # Beginning of art topic definition
top-level topic       art ACCRUE
topic definition modifiers
                          /author = "fsmith"
                          /date   = "30-Dec-01"
                          /annotation = "Topic created
                                         by fsmith"
subtopic topic        * 0.70 performing-arts ACCRUE
  evidence topic      ** 0.50 WORD
  topic definition modifier
                          /wordtext = ballet
  evidence topic      ** 0.50 STEM
  topic definition modifier
                          /wordtext = dance
  evidence topic      ** 0.50 WORD
  topic definition modifier
                          /wordtext = opera
  evidence topic      ** 0.30 WORD
  topic definition modifier
                          /wordtext = symphony
subtopic              * 0.70 visual-arts ACCRUE
                      ** 0.50 WORD
                          /wordtext = painting
                      ** 0.50 WORD
                          /wordtext = sculpture
subtopic              * 0.70 film ACCRUE
                      ** 0.50 STEM
                          /wordtext = film
subtopic              ** 0.50 motion-picture PHRASE
                      *** 1.00 WORD
                          /wordtext = motion
                      *** 1.00 WORD
                          /wordtext = picture
                      ** 0.50 STEM
                          /wordtext = movie
subtopic              * 0.50 video ACCRUE
                      ** 0.50 STEM
                          /wordtext = video
                      ** 0.50 STEM
                          /wordtext = vcr
                      # End of art topic
```

- Note:
  - maintenance issues (author, etc.)
  - Hand-weighting of terms

[Verity was bought by Autonomy, which was bought by HP ...]

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) – see video
  - k-Nearest Neighbors (simple, powerful)
  - Support-vector machines (new, generally more powerful)
  - … plus many other methods
  - No free lunch: requires hand-classified training data
  - But data can be built up (and refined) by amateurs
- Many commercial systems use a mixture of methods

The bag of words representation

$\gamma($ I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun… It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet. $) = c$

The bag of words representation

Features

- Supervised learning classifiers can use any sort of feature
  - URL, email address, punctuation, capitalization, dictionaries, network features
- In the bag of words view of documents
  - We use **only** word features
  - we use **all** of the words in the text (not a subset)

Feature Selection: Why?

- Text collections have a large number of features
  - 10,000 – 1,000,000 unique words … and more
- Selection may make a particular classifier feasible
  - Some classifiers can't deal with 1,000,000 features
- Reduces training time
  - Training time for some methods is quadratic or worse in the number of features
- Makes runtime models smaller and faster
- Can improve generalization (performance)
  - Eliminates noise features
  - Avoids overfitting

Feature Selection: Frequency

- The simplest feature selection method:
  - Just use the most common terms
  - No particular foundation
  - But it make sense why this works
    - They're the words that can be well-estimated and are most often available as evidence
  - In practice, this is often 90% as good as better methods
  - Smarter feature selection – future lecture

Quiz: Bag of Words

- Why don't we use sequence of words as the representation for documents when doing text classification for IR?

Evaluating Categorization

- Evaluation must be done on test data that are independent of the training data
  - Sometimes use cross-validation (averaging results over multiple training and test splits of the overall data)
- Easy to get good performance on a test set that was available to the learner during training (e.g., just memorize the test set)

Evaluating Categorization

- Measures: precision, recall, F1, classification accuracy
- Classification accuracy: r/n where n is the total number of test docs and r is the number of test docs correctly classified

WebKB Experiment (1998)

- Classify webpages from CS departments into:
  - student, faculty, course, project
- Train on ~5,000 hand-labeled web pages
  - Cornell, Washington, UTexas, Wisconsin
- Crawl and classify a new site (CMU) using Naïve Bayes

- Results

	Student	Faculty	Person	Project	Course	Departmt
Extracted	180	66	246	99	28	1
Correct	130	28	194	72	25	1
Accuracy:	72%	42%	79%	73%	89%	100%

Faculty

associate	0.00417
chair	0.00303
member	0.00288
ph	0.00287
director	0.00282
fax	0.00279
journal	0.00271
recent	0.00260
received	0.00258
award	0.00250

Students

resume	0.00516
advisor	0.00456
student	0.00387
working	0.00361
stuff	0.00359
links	0.00355
homepage	0.00345
interests	0.00332
personal	0.00332
favorite	0.00310

Courses

homework	0.00413
syllabus	0.00399
assignments	0.00388
exam	0.00385
grading	0.00381
midterm	0.00374
pm	0.00371
instructor	0.00370
due	0.00364
final	0.00355

Departments

departmental	0.01246
colloquia	0.01076
epartment	0.01045
seminars	0.00997
schedules	0.00879
webmaster	0.00879
events	0.00826
facilities	0.00807
eople	0.00772
postgraduate	0.00764

Research Projects

investigators	0.00256
group	0.00250
members	0.00242
researchers	0.00241
laboratory	0.00238
develop	0.00201
related	0.00200
arpa	0.00187
affiliated	0.00184
project	0.00183

Others

type	0.00164
jan	0.00148
enter	0.00145
random	0.00142
program	0.00136
net	0.00128
time	0.00128
format	0.00124
access	0.00117
begin	0.00116

SpamAssassin

- Naïve Bayes has found a home in spam filtering
  - Paul Graham's A Plan for Spam
  - Widely used in spam filters
  - But many features beyond words:
    - black hole lists, etc.
    - particular hand-crafted text patterns

SpamAssassin Features:

- Basic (Naïve) Bayes spam probability
- Mentions: Generic Viagra
- Regex: millions of (dollar) ((dollar) NN,NNN,NNN.NN)
- Phrase: impress ... girl
- Phrase: 'Prestigious Non-Accredited Universities'
- From: starts with many numbers
- Subject is all capitals
- HTML has a low ratio of text to image area
- Relay in RBL (realtime blackhole list), http://www.mail-abuse.com/enduserinfo_rbl.html
- RCVD line looks faked
- http://spamassassin.apache.org/tests_3_3_x.html

Naive Bayes is Not So Naive

- Very fast learning and testing (basically just count words)
- Low storage requirements
- Very good in domains with many equally important features
- More robust to irrelevant features than many learning methods

  Irrelevant features cancel each other without affecting results

Naive Bayes is Not So Naive

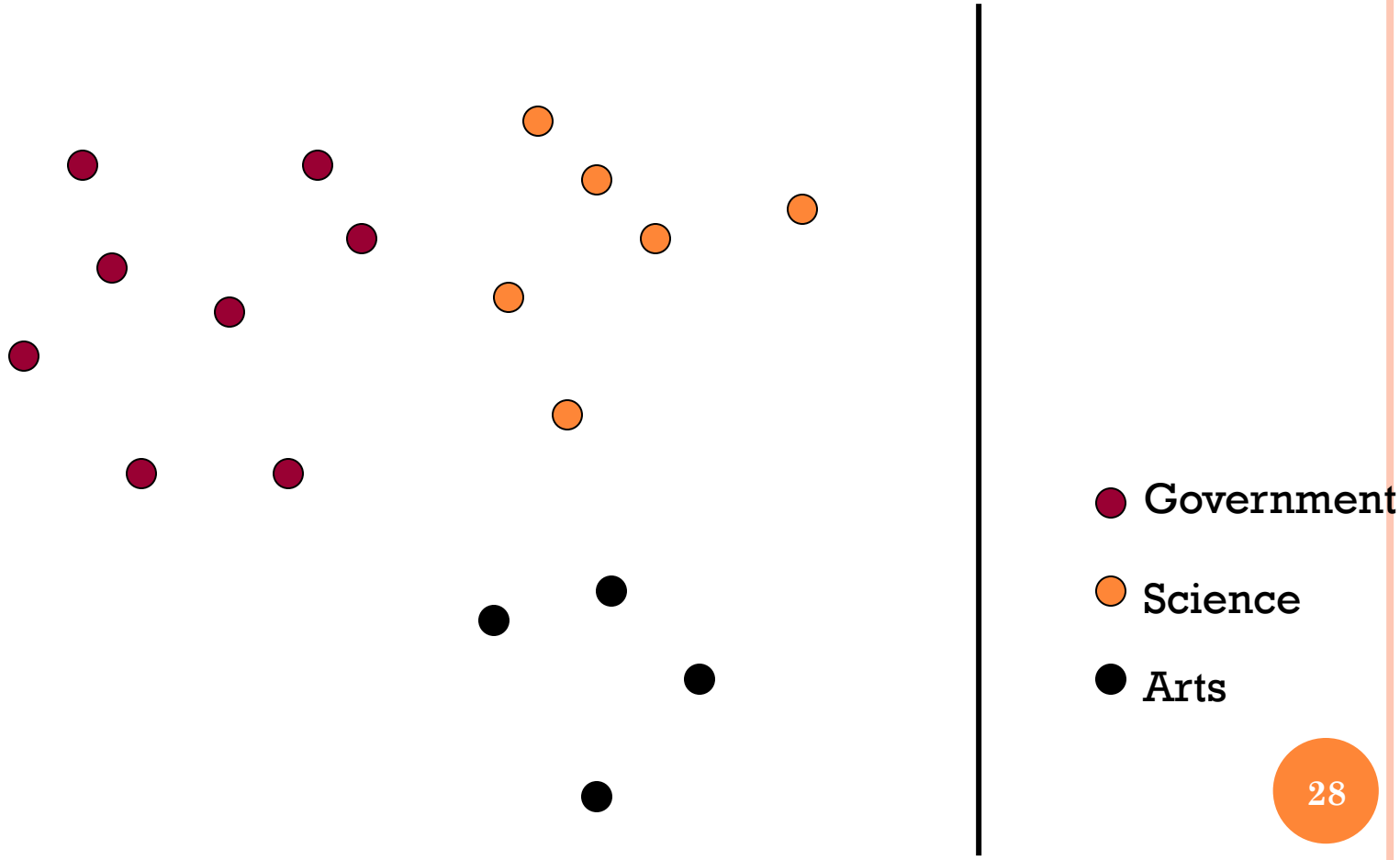
- More robust to concept drift (changing class definition over time)
- Naive Bayes won 1st and 2nd place in KDD-CUP 97 competition out of 16 systems

  Goal: Financial services industry direct mail response prediction: Predict if the recipient of mail will actually respond to the advertisement – 750,000 records.
- A good dependable baseline for text classification (but not the best)!

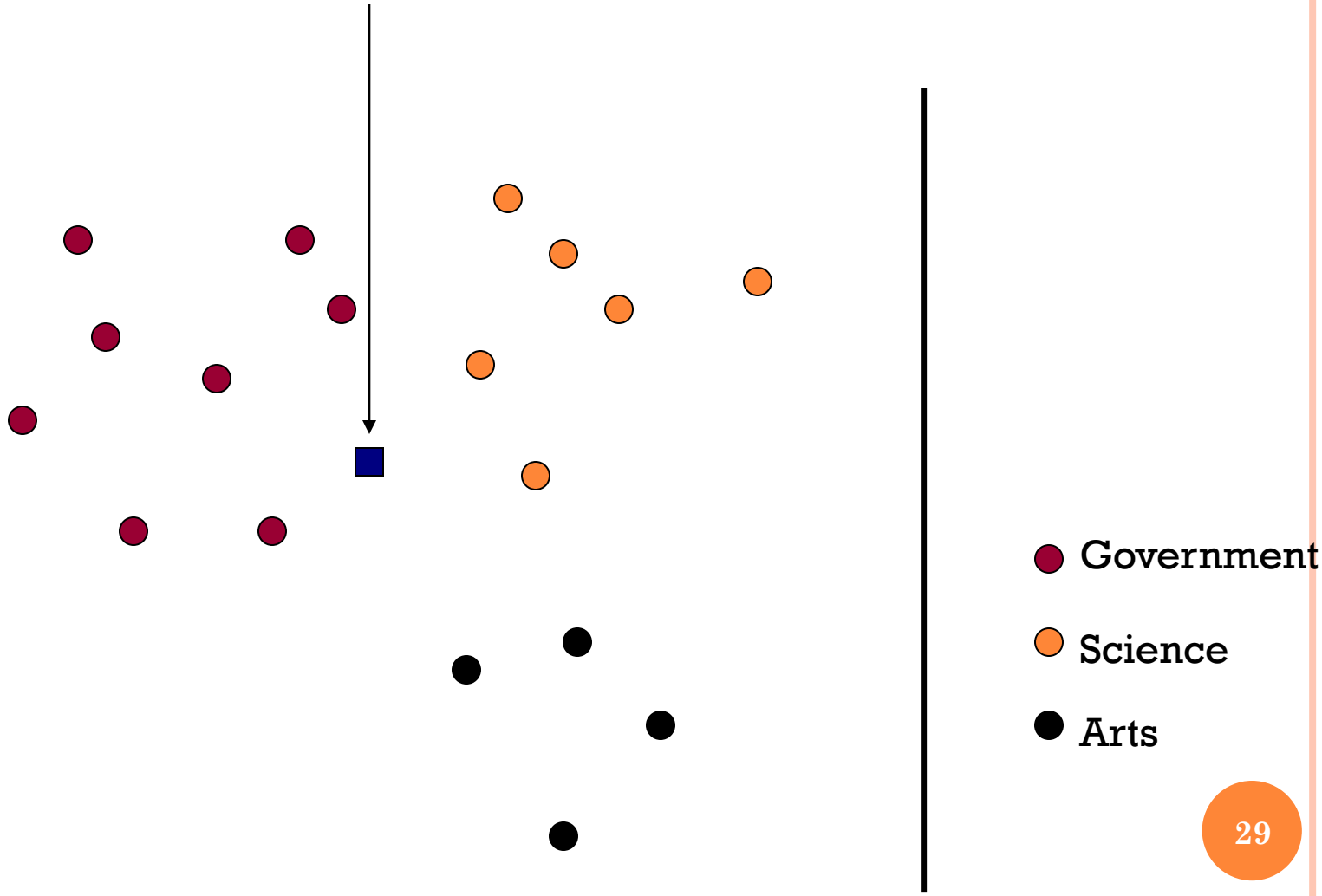
Classification Using Vector Spaces

- 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

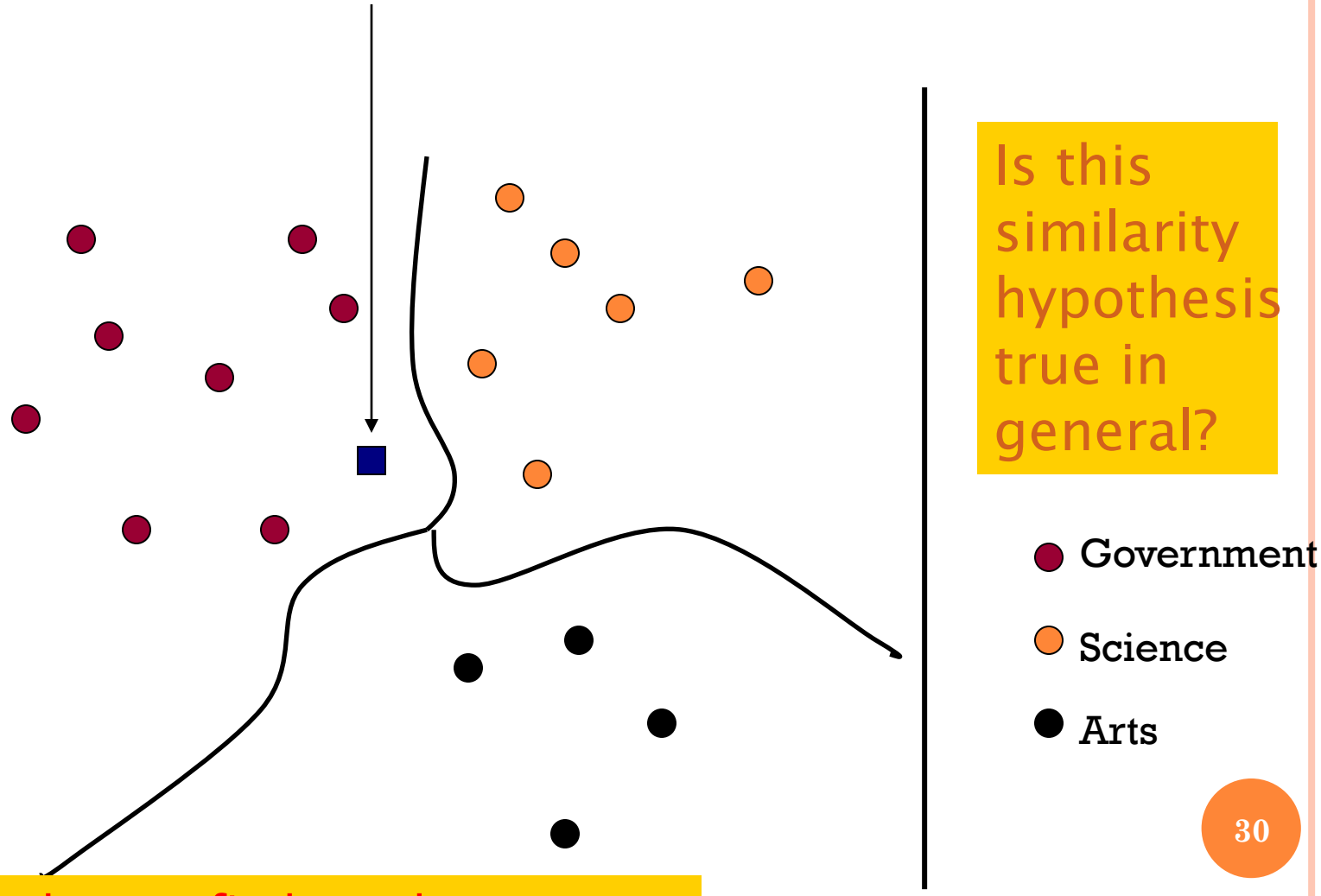
Documents in a Vector Space



Test Document of what class?



Test Document = Government



Our focus: how to find good separators

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.

- *Note that centroid will in general* not *be a unit vector even when the inputs are unit vectors.*

Rocchio classification

- Rocchio forms a simple representative for each class: the centroid/prototype
- Classification: **nearest prototype/centroid**
- It does not guarantee that classifications are consistent with the given training data

Rocchio classification

- Little used outside text classification
  - It has been used quite effectively for text classification
  - But in general worse than Naïve Bayes
- Again, cheap to train and test documents

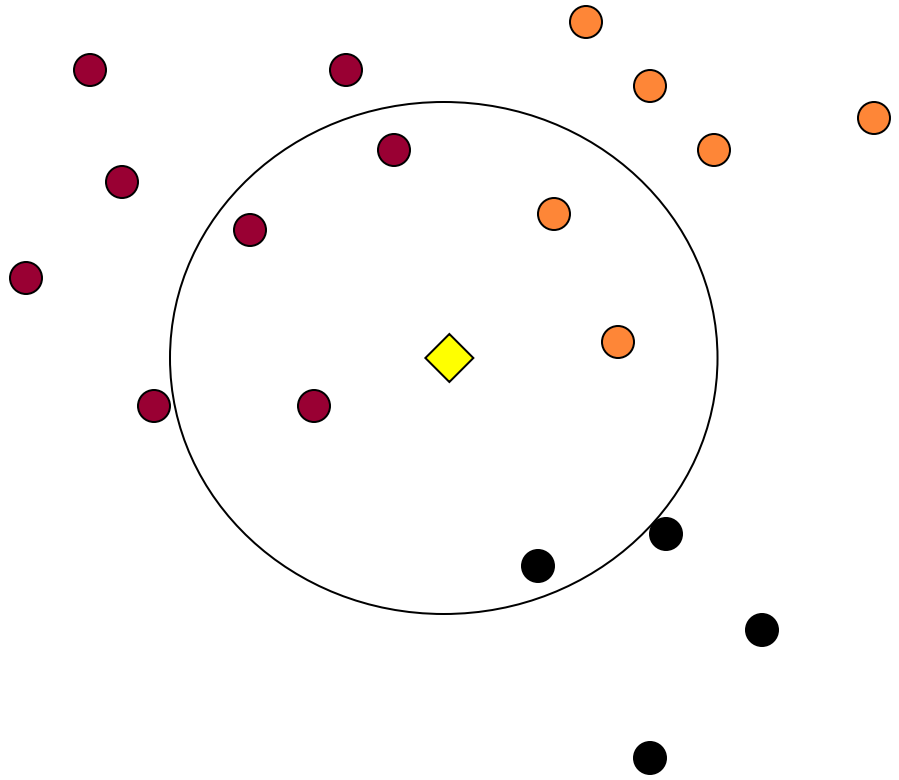
K Nearest Neighbor Classification

- kNN = *k* Nearest Neighbor

- To classify a document *d*:
- Define *k*-neighborhood as the *k* nearest neighbors of *d*
- Pick the majority class label in the *k*-neighborhood

QUIZ: κ=6 (6NN)

- What is the class of ◆?



Government

Science

Arts

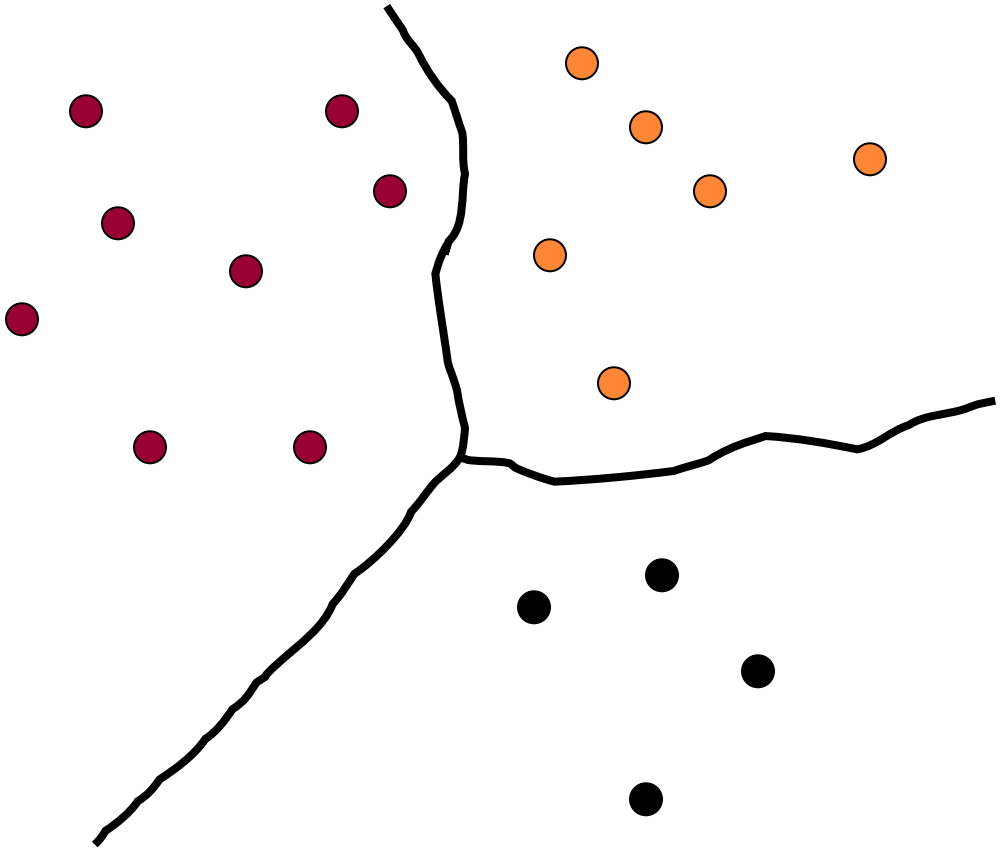
Nearest-Neighbor Learning

- Learning: just store the labeled training examples *D*
- Testing instance *x (under 1NN)*:
  - Compute similarity between *x* and all examples in *D*.
  - Assign *x* the category of the most similar example in *D*.
- Does not compute anything beyond storing the examples
- Also called:
  - Case-based learning
  - Memory-based learning
  - Lazy learning
- Rationale of kNN: contiguity hypothesis

K Nearest Neighbor

- Using only the closest example (1NN) subject to errors due to:
  - A single atypical example.
  - Noise (i.e., an error) in the category label of a single training example.
- More robust: find the *k* examples and return the majority category of these *k*
- *k* is typically odd to avoid ties; 3 and 5 are most common

kNN DECISION BOUNDARIES

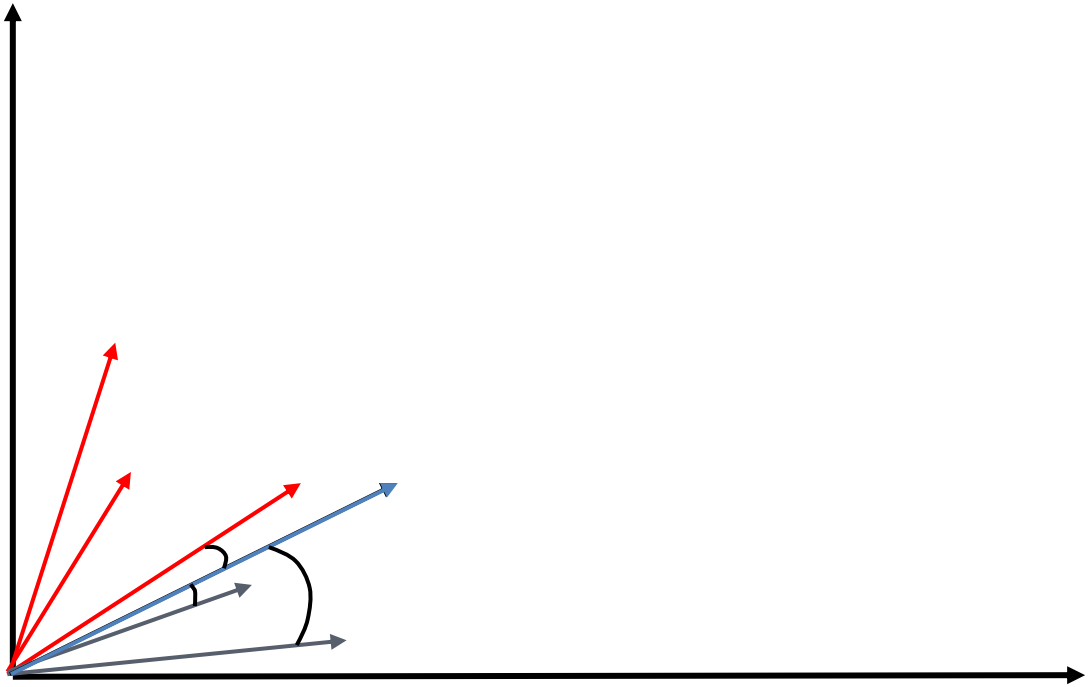


Boundaries are in principle arbitrary surfaces – but usually polyhedra

- Government
- Science
- Arts

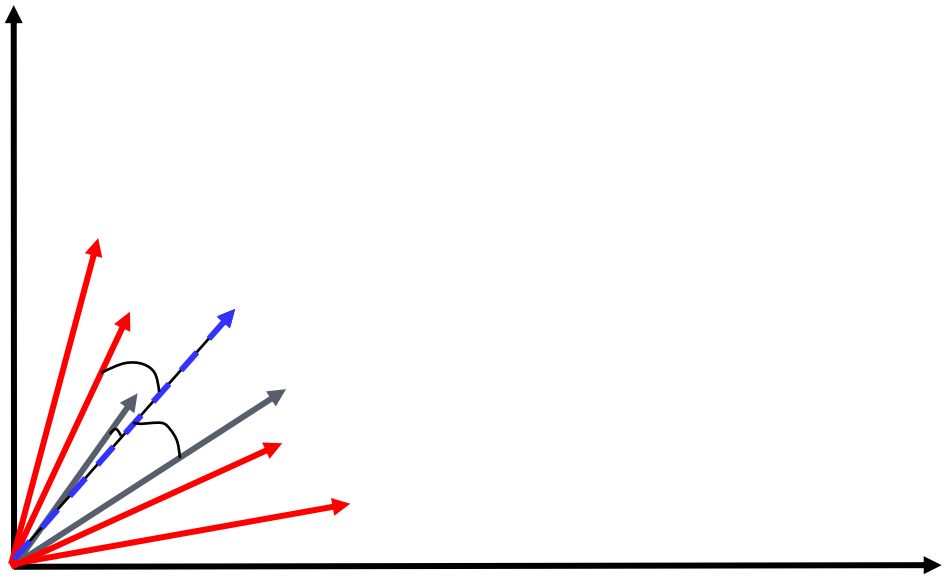
kNN gives locally defined decision boundaries between classes – far away points do not influence each classification decision (unlike in Naïve Bayes, Rocchio, etc.)

Illustration of 3 Nearest Neighbor for Text Vector Space



3 Nearest Neighbor vs. Rocchio

- Nearest Neighbor tends to handle polymorphic categories better than Rocchio/NB.



kNN: Discussion

- No feature selection necessary
- No training necessary
- Scales well with large number of classes
  - Don't need to train *n* classifiers for *n* classes
- Classes can influence each other
  - Small changes to one class can have ripple effect
- May be expensive at test time
- In most cases it's more accurate than NB or Rocchio

Let's test our intuition

- Can a bag of words always be viewed as a vector space?
- What about a bag of features?
- Can we always view a standing query as a region in a vector space?
- What about Boolean queries on terms?
- What do "rectangles" equate to?

Bias vs. capacity – notions and terminology

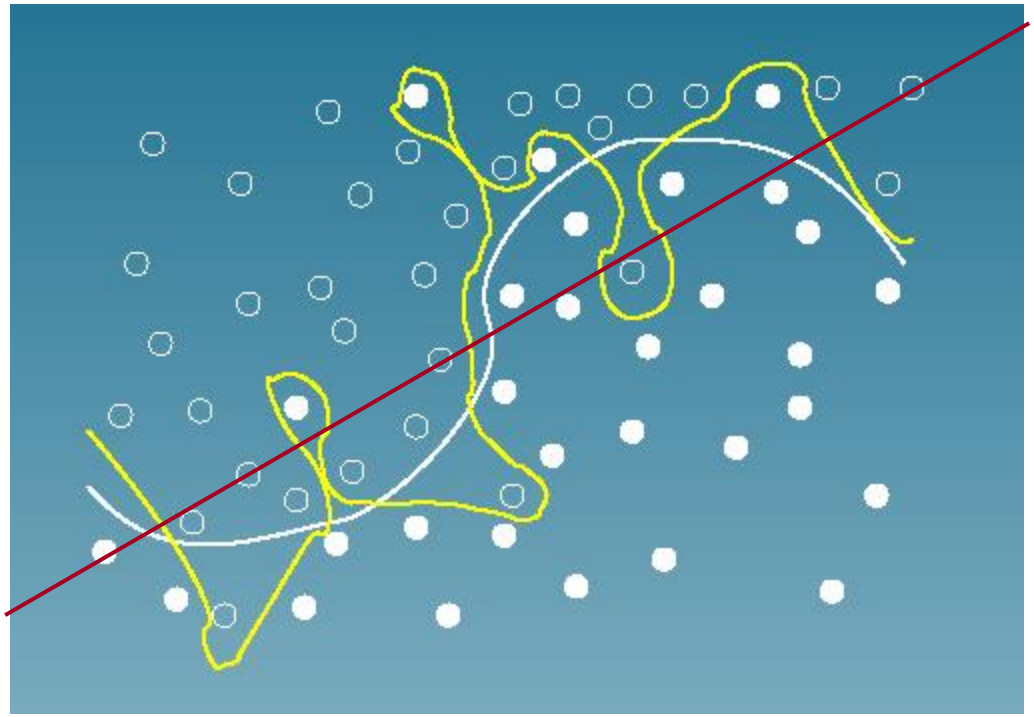
- Consider asking a botanist: Is an object a tree?
  - Too much *capacity (details)*, low *bias*
    - Botanist who memorizes
    - Will always say "no" to new object (e.g., different # of leaves)
  - Not enough capacity, high bias (high abstraction)
    - Lazy botanist
    - Says "yes" if the object is green
  - You want the middle ground

(Example due to C. Burges)

kNN vs. Naive Bayes

- Bias/Variance tradeoff
  - Variance ≈ Capacity
- kNN has high variance and low bias.
  - Infinite memory
- NB has low variance and high bias.
  - Linear decision surface (hyperplane – SVM)

Bias vs. variance: Choosing the correct model capacity

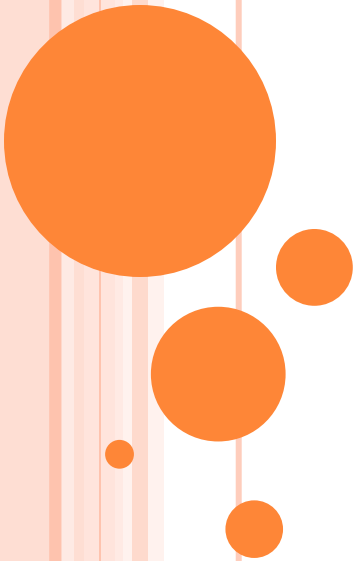


Summary: Representation of Text Categorization Attributes

- Representations of text are usually very high dimensional
- High-bias algorithms that prevent overfitting should generally work best in high-dimensional space
- For most text categorization tasks, there are many relevant features and many irrelevant ones

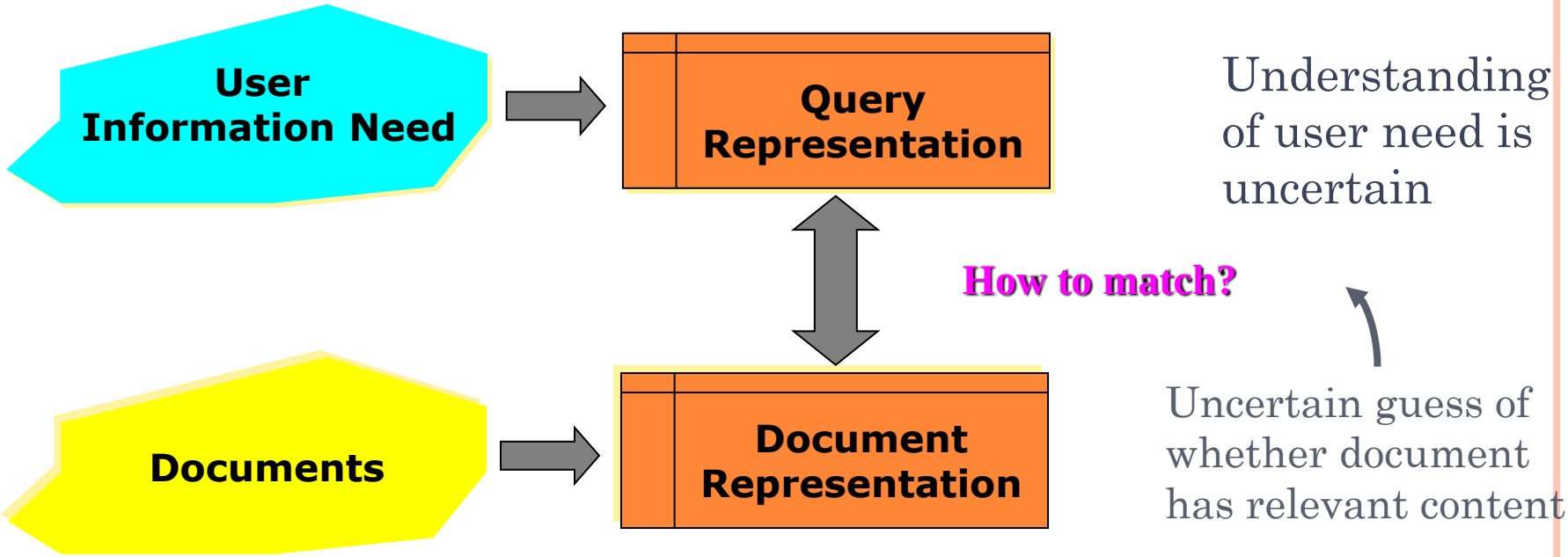
Which classifier do I use for a given text classification problem?

- Is there a learning method that is optimal for all text classification problems?
- No, because there is a tradeoff between bias and variance.
- Factors to take into account:
  - How much training data is available?
  - How simple/complex is the problem? (linear vs. nonlinear decision boundary)
  - How noisy is the data?
  - How stable is the problem over time?
    - For an unstable problem, its better to use a simple and robust classifier.



Probabilistic Information Retrieval

Why probabilities in IR?



In traditional IR systems, matching between each document and query is attempted in a semantically imprecise space of index terms.

Probabilities provide a principled foundation for uncertain reasoning.
Can we use probabilities to quantify our uncertainties?

Probabilistic IR topics

- **Classical probabilistic retrieval model**
  - Probability ranking principle, etc.
- (Naïve) Bayesian Text Categorization
- Bayesian networks for text retrieval
- **Language model approach to IR**
  - An important emphasis in recent work

We present these two!

- *Probabilistic methods are one of the oldest but also one of the currently hottest topics in IR.*
  - *Traditionally: neat ideas, but they've never won on performance. It may be different now.*

The document ranking problem

- We have a collection of documents
- User issues a query
- A list of documents needs to be returned
- **Ranking method is core of an IR system:**
  - **In what order do we present documents to the user?**
  - We want the "best" document to be first, second best second, etc….
- **Idea: Rank by probability of relevance of the document w.r.t. information need**
  - P(relevant | document_i, query)

Recall a few probability basics

- For events a and b:
- Bayes' Rule

$$p(a,b) = p(a \cap b) = p(a \mid b)\,p(b) = p(b \mid a)\,p(a)$$

$$p(\bar{a} \mid b)\,p(b) = p(b \mid \bar{a})\,p(\bar{a})$$

$$p(a \mid b) = \frac{p(b \mid a)\,p(a)}{p(b)} = \frac{p(b \mid a)\,p(a)}{\sum_{x=a,\bar{a}} p(b \mid x)\,p(x)}$$

Prior

Posterior

- Odds:

$$O(a) = \frac{p(a)}{p(\bar{a})} = \frac{p(a)}{1 - p(a)}$$

The Probability Ranking Principle

"If a reference retrieval system's response to each request is a ranking of the documents in the collection in order of decreasing probability of relevance to the user who submitted the request, where the probabilities are estimated as accurately as possible on the basis of **whatever data have been made available** to the system for this purpose, the overall effectiveness of the system to its user will be the best that is obtainable on the basis of those data."

- [1960s/1970s] S. Robertson, W.S. Cooper, M.E. Maron; van Rijsbergen (1979:113); Manning & Schütze (1999:538)

Probability Ranking Principle

Let x be a document in the collection.
Let R represent **relevance** of a document w.r.t. given (fixed) query and let NR represent **non-relevance.**

R={0,1} vs. NR/R

Need to find $p(R|x)$ - probability that a document x is **relevant.**

$$p(R \mid x) = \frac{p(x \mid R)\,p(R)}{p(x)}$$

$$p(NR \mid x) = \frac{p(x \mid NR)\,p(NR)}{p(x)}$$

$p(R)$,$p(NR)$ - prior probability of retrieving a (non) relevant document

$$p(R \mid x) + p(NR \mid x) = 1$$

$p(x|R)$, $p(x|NR)$ - probability that if a relevant (non-relevant) document is retrieved, it is x.

Probability Ranking Principle (PRP)

- Simple case: no selection costs or other utility concerns that would differentially weight errors
- ***Bayes' Optimal Decision Rule***
  - x is **relevant** iff $p(R \mid x) > p(NR \mid x)$
- PRP in action: Rank all documents by $p(R \mid x)$
- Theorem:
  - Using the PRP is optimal, in that it minimizes the loss (Bayes risk) under 1/0 loss
  - Provable if all probabilities correct, etc. [e.g., Ripley 1996]

Probability Ranking Principle

- More complex case: retrieval costs.
  - Let *d* be a document
  - *C* - cost of retrieval of relevant document
  - *C'* - cost of retrieval of non-relevant document
- Probability Ranking Principle: if

$$C \cdot p(R|d) + C' \cdot (1 - p(R|d)) \leq C \cdot p(R|d') + C' \cdot (1 - p(R|d'))$$

for all *d' not yet retrieved*, then *d* **is the next document to be retrieved**

- **We won't further consider loss/utility from now on**

Probability Ranking Principle

- How do we compute all those probabilities?
  - Do not know exact probabilities, have to use estimates
  - Binary Independence Retrieval (BIR) – which we discuss later today – is the simplest model
- Questionable assumptions
  - "Relevance" of each document is independent of relevance of other documents.
    - Really, it's bad to keep on returning **duplicates**
  - Boolean model of relevance
  - That one has a single step information need
    - Seeing a range of results might let user refine query

Probabilistic Retrieval Strategy

- Estimate how terms contribute to relevance
  - How do things like tf, df, and length influence your judgments about document relevance?
    - One answer is the Okapi BM25 (S. Robertson)
- Combine to find document relevance probability
- Order documents by decreasing probability

Probabilistic Ranking

Basic concept:

"For a given query, if we know some documents that are relevant, terms that occur in those documents should be given greater weighting in searching for other relevant documents.

By making assumptions about the distribution of terms and applying Bayes Theorem, it is possible to derive weights theoretically."

Van Rijsbergen

Binary Independence Model

- Traditionally used in conjunction with PRP
- **"Binary" = Boolean**: documents are represented as binary incidence vectors of terms (cf. lecture 1):
  - $\vec{x} = (x_1, \ldots, x_n)$
  - $x_i = 1$ iff term i is present in document x.
- **"Independence":** terms occur in documents independently
- Different documents can be modeled as same vector

- Bernoulli Naive Bayes model (cf. text categorization!)

Binary Independence Model

- Queries: binary term incidence vectors
- Given query q,
  - for each document d need to compute $p(R|q,d)$.
  - replace with computing $p(R|q,x)$ where x is binary term incidence vector representing d
  - Interested only in ranking
- Will use odds and Bayes' Rule:

$$O(R|q,\vec{x}) = \frac{p(R|q,\vec{x})}{p(NR|q,\vec{x})} = \frac{\frac{p(R|q)\,p(\vec{x}|R,q)}{p(\vec{x}|q)}}{\frac{p(NR|q)\,p(\vec{x}|NR,q)}{p(\vec{x}|q)}}$$

Binary Independence Model

$$O(R \mid q, \vec{x}) = \frac{p(R \mid q, \vec{x})}{p(NR \mid q, \vec{x})} = \frac{p(R \mid q)}{p(NR \mid q)} \cdot \frac{p(\vec{x} \mid R, q)}{p(\vec{x} \mid NR, q)}$$

Constant for a given query

Needs estimation

- Using **Independence** Assumption:

$$\frac{p(\vec{x} \mid R, q)}{p(\vec{x} \mid NR, q)} = \prod_{i=1}^{n} \frac{p(x_i \mid R, q)}{p(x_i \mid NR, q)}$$

- So : $$O(R \mid q, d) = O(R \mid q) \cdot \prod_{i=1}^{n} \frac{p(x_i \mid R, q)}{p(x_i \mid NR, q)}$$

Binary Independence Model

$$O(R \mid q,d) = O(R \mid q) \cdot \prod_{i=1}^{n} \frac{p(x_i \mid R,q)}{p(x_i \mid NR,q)}$$

- Since x_i is either 0 or 1:

$$O(R \mid q,d) = O(R \mid q) \cdot \prod_{x_i=1} \frac{p(x_i=1 \mid R,q)}{p(x_i=1 \mid NR,q)} \cdot \prod_{x_i=0} \frac{p(x_i=0 \mid R,q)}{p(x_i=0 \mid NR,q)}$$

- Let $p_i = p(x_i=1 \mid R,q); \quad r_i = p(x_i=1 \mid NR,q);$

- Assume, for all terms not occurring in the query ($q_i=0$) $p_i = r_i$

Then...

This can be changed (e.g., in relevance feedback)

Binary Independence Model

$$O(R \mid q, \vec{x}) = O(R \mid q) \cdot \prod_{x_i = q_i = 1} \frac{p_i}{r_i} \cdot \prod_{\substack{x_i = 0 \\ q_i = 1}} \frac{1 - p_i}{1 - r_i}$$

All matching terms

Non-matching query terms

$$= O(R \mid q) \cdot \prod_{x_i = q_i = 1} \frac{p_i (1 - r_i)}{r_i (1 - p_i)} \cdot \prod_{q_i = 1} \frac{1 - p_i}{1 - r_i}$$

All matching terms

All query terms

Binary Independence Model

$$O(R \mid q, \vec{x}) = O(R \mid q) \cdot \prod_{x_i = q_i = 1} \frac{p_i (1 - r_i)}{r_i (1 - p_i)} \cdot \prod_{q_i = 1} \frac{1 - p_i}{1 - r_i}$$

Constant for each query

Only quantity to be estimated for rankings

- **Retrieval Status Value**:

$$RSV = \log \prod_{x_i = q_i = 1} \frac{p_i (1 - r_i)}{r_i (1 - p_i)} = \sum_{x_i = q_i = 1} \log \frac{p_i (1 - r_i)}{r_i (1 - p_i)}$$

Binary Independence Model

- All boils down to computing RSV.

$$RSV = \log \prod_{x_i = q_i = 1} \frac{p_i(1-r_i)}{r_i(1-p_i)} = \sum_{x_i = q_i = 1} \log \frac{p_i(1-r_i)}{r_i(1-p_i)}$$

$$RSV = \sum_{x_i = q_i = 1} c_i; \qquad c_i = \log \frac{p_i(1-r_i)}{r_i(1-p_i)}$$

So, how do we compute *ci's* from our data ?

Binary Independence Model

- Estimating RSV coefficients.
- For each term *i* look at this table of document counts:

Document	Relevant	Non-Relevant	Total
$X_i=1$	s	$n-s$	n
$X_i=0$	$S-s$	$N-n-S+s$	$N-n$
Total	S	$N-S$	N

This is df_i!

- Estimates: $p_i \approx \frac{s}{S}$ $r_i \approx \frac{(n-s)}{(N-S)}$

$$c_i \approx K(N,n,S,s) = \log \frac{s/(S-s)}{(n-s)/(N-n-S+s)}$$

Estimation – key challenge

- If non-relevant documents are approximated by the whole collection, then r_i (prob. of occurrence in non-relevant documents for query) *is* n/N and
  - $\log (1- r_i)/r_i = \log (N- n)/n \cong \log N/n$ = IDF!
- p_i (probability of occurrence in relevant documents) can be estimated in various ways:
  - from relevant documents if know some
    - Relevance weighting can be used in feedback loop
  - constant (Croft and Harper combination match) – then just get idf weighting of terms
  - proportional to prob. of occurrence in collection
    - more accurately, the log of this (Greiff, SIGIR 1998)

Iteratively estimating P_I

1. Assume that p_i constant over all x_i in query
   - $p_i = 0.5$ (even odds) for any given doc
2. Determine a guess of relevant document set:
   - V is fixed size set of highest ranked documents on this model (note: now a bit like tf.idf!)
3. We need to improve our guesses for p_i and r_i, so
   - Use distribution of x_i in docs in V. Let V_i be set of documents containing x_i
     - $p_i = |V_i| / |V|$
   - Assume if not retrieved then not relevant
     - $r_i = (n_i - |V_i|) / (N - |V|)$
4. Go to 2. until converges then return ranking

PRP and BIR

- Getting reasonable approximations of probabilities is possible.
- Requires restrictive assumptions:
  - ***term independence***
  - ***terms not in query don't affect the outcome***
  - ***boolean representation of documents/queries/relevance***
  - ***document relevance values are independent***
- Some of these assumptions can be removed
- Problem: either require partial relevance information or only can derive somewhat inferior term weights

Resources

S. E. Robertson and K. Spärck Jones. 1976. Relevance Weighting of Search Terms. *Journal of the American Society for Information Sciences* 27(3): 129–146.

C. J. van Rijsbergen. 1979. *Information Retrieval*. 2nd ed. London: Butterworths, chapter 6. [Most details of math] http://www.dcs.gla.ac.uk/Keith/Preface.html

N. Fuhr. 1992. Probabilistic Models in Information Retrieval. *The Computer Journal*, 35(3),243–255. [Easiest read, with BNs]

F. Crestani, M. Lalmas, C. J. van Rijsbergen, and I. Campbell. 1998. Is This Document Relevant? ... Probably: A Survey of Probabilistic Models in Information Retrieval. *ACM Computing Surveys* 30(4): 528–552.

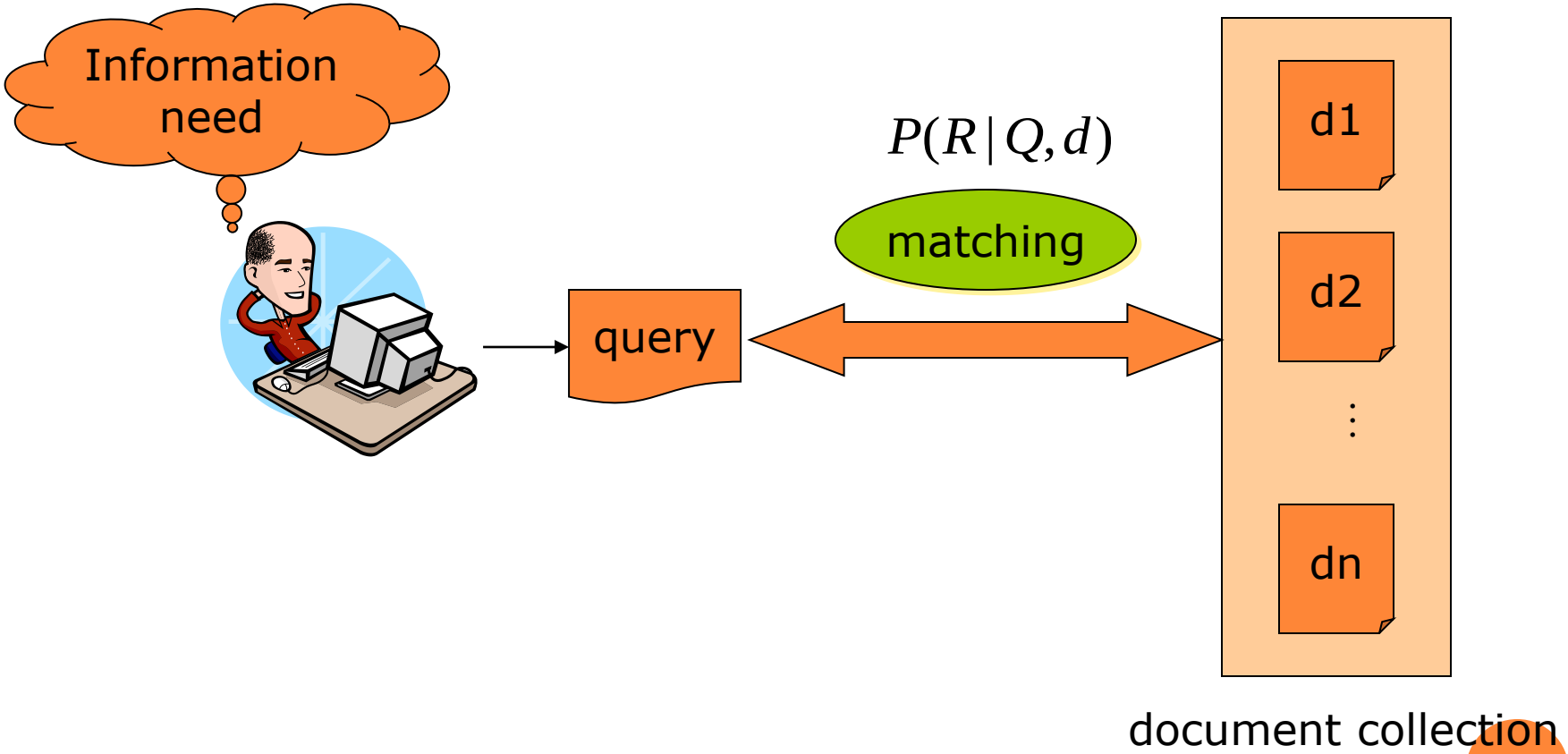
http://www.acm.org/pubs/citations/journals/surveys/1998-30-4/p528-crestani/

[Adds very little material that isn't in van Rijsbergen or Fuhr]

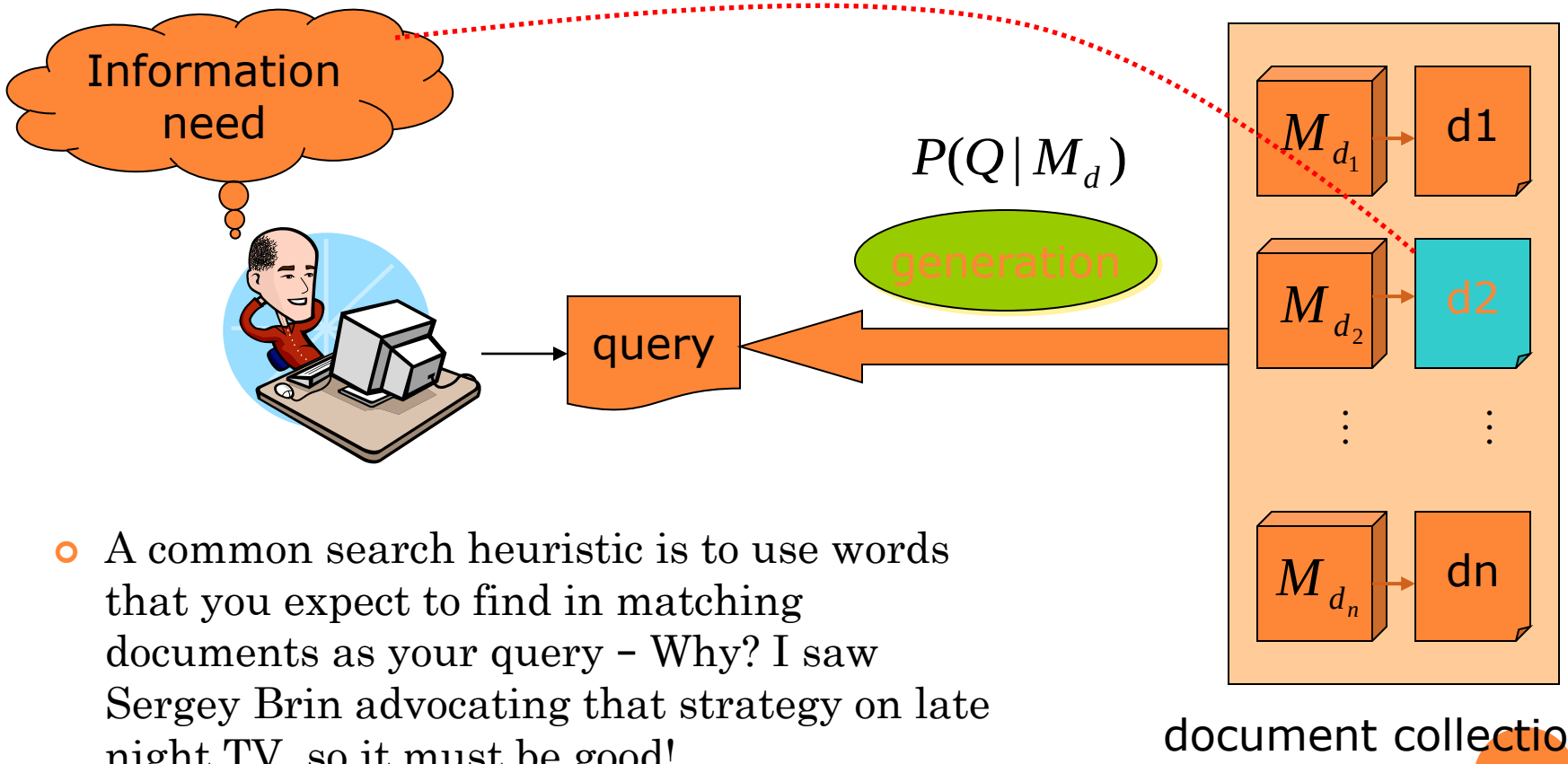
Language Model

- The Language Model Approach to IR
  - Basic query generation model
  - Alternative models

Standard Probabilistic IR



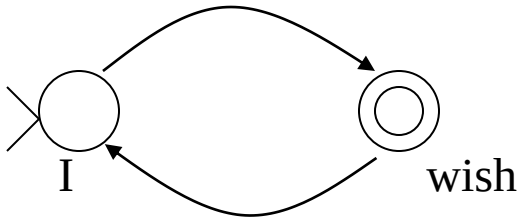
IR BASED ON LANGUAGE MODEL (LM)



- A common search heuristic is to use words that you expect to find in matching documents as your query – Why? I saw Sergey Brin advocating that strategy on late night TV, so it must be good!
- The LM approach directly exploits that idea!

Formal Language (Model)

- Traditional generative model: generates strings
  - Finite state machines or regular grammars, etc.
- Example:



I wish

I wish I wish

I wish I wish I wish

I wish I wish I wish I wish

…

I wish (I wish)*

Stochastic Language Models

- Models *probability* of generating strings in the language (commonly all strings over alphabet ∑)

Model M

0.2	the
0.1	a
0.01	man
0.01	woman
0.03	said
0.02	likes
…	

the	man	likes	the	woman
0.2	0.01	0.02	0.2	0.01

multiply

P(s | M) = 0.00000008

Stochastic Language Models

- Model *probability* of generating any string

Model M1	
0.2	the
0.01	class
0.0001	sayst
0.0001	pleaseth
0.0001	yon
0.0005	maiden
0.01	woman

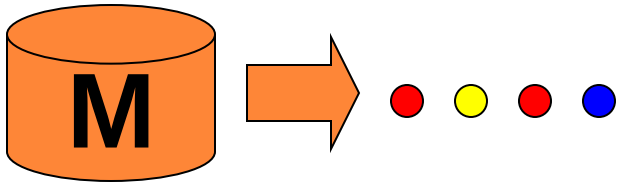
Model M2	
0.2	the
0.0001	class
0.03	sayst
0.02	pleaseth
0.1	yon
0.01	maiden
0.0001	woman

the	class	pleaseth	yon	maiden
0.2	0.01	0.0001	0.0001	0.0005
0.2	0.0001	0.02	0.1	0.01

P(s|M2) > P(s|M1)

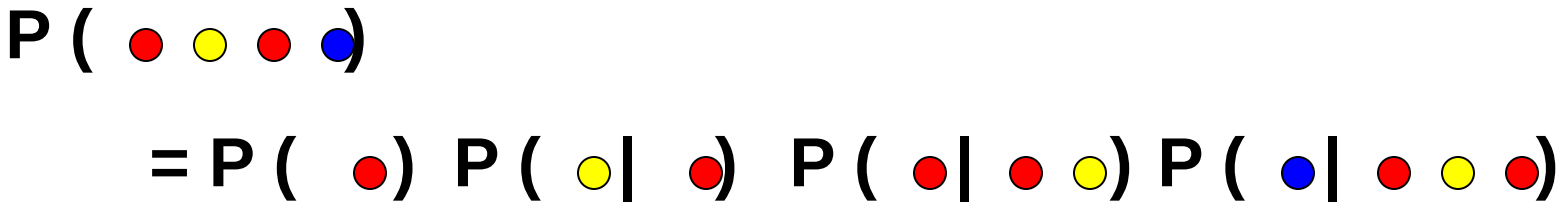
Stochastic Language Models

- A statistical model for generating text
  - Probability distribution over strings in a given language



P(● ○ ● ● | M) = P(● | M)
P(○ | M, ●)
P(● | M, ● ○)
P(● | M, ● ○ ●)

Unigram and higher-order models



- Unigram Language Models

Easy.
Effective!

- Bigram (generally, *n*-gram) Language Models

- Other Language Models
  - Grammar-based models (PCFGs), etc.
    - Probably not the first thing to try in IR

Using Language Models in IR

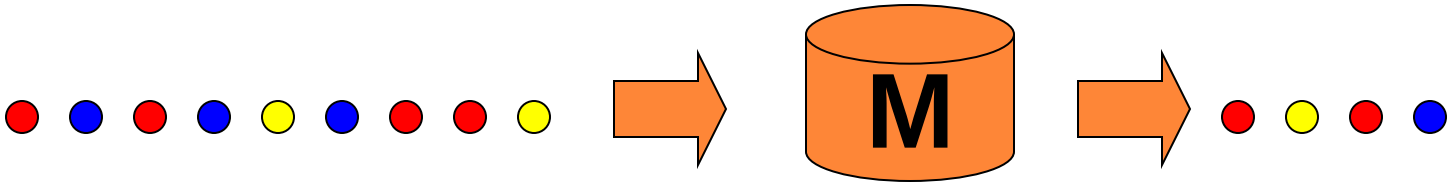
- Treat each document as the basis for a model (e.g., unigram sufficient statistics)
- Rank document d based on P(d | q)
- P(d | q) = P(q | d) x P(d) / P(q)
  - P(q) is the same for all documents, so ignore
  - P(d) [the prior] is often treated as the same for all d
    - But we could use criteria like authority, length, genre
  - P(q | d) is the probability of q given d's model
- Very general formal approach

The fundamental problem of LMs

- Usually we don't know the model **M**
  - But have a sample of text representative of that model

P(● ● ● ● | M(● ● ● ● ● ● ● ● ● ●))

- Estimate a language model from a sample
- Then compute the observation probability



Language Models for IR

- Language Modeling Approaches
  - Attempt to **model query generation process**
  - Documents are ranked by **the probability that a query would be observed as a random sample from the respective document model**
    - Multinomial approach

$$P(Q|M_D) = \prod_w P(w|M_D)^{q_w}$$

Retrieval based on probabilistic LM

- Treat the generation of queries as a random process.
- Approach
  - Infer a language model for each document.
  - Estimate the probability of generating the query according to each of these models.
  - Rank the documents according to these probabilities.
  - Usually a unigram estimate of words is used
    - Some work on bigrams, paralleling van Rijsbergen

Retrieval based on probabilistic LM

- Intuition
  - Users …
    - Have a reasonable idea of terms that are likely to occur in documents of interest.
    - They will choose query terms that distinguish these documents from others in the collection.
- Collection statistics …
  - Are integral parts of the language model.
  - Are not used heuristically as in many other approaches.
    - In theory. In practice, there's usually some wiggle room for empirically set parameters

Query generation probability (1)

- Ranking formula $$p(Q,d) = p(d)\,p(Q \mid d)$$
$$\approx p(d)\,p(Q \mid M_d)$$

- The probability of producing the query given the language model of document d using MLE is:

$$\hat{p}(Q \mid M_d) = \prod_{t \in Q} \hat{p}_{ml}(t \mid M_d)$$

$$= \prod_{t \in Q} \frac{tf_{(t,d)}}{dl_d}$$

Unigram assumption:
Given a particular language model, the query terms occur independently

M_d : language model of document d

$tf_{(t,d)}$: raw tf of term t in document d

dl_d : total number of tokens in document d

Insufficient data

- Zero probability $p(t \mid M_d) = 0$
  - May not wish to assign a probability of zero to a document that is missing one or more of the query terms [gives conjunction semantics]
- General approach
  - A non-occurring term is possible, but no more likely than would be expected by chance in the collection.
  - If $tf_{(t,d)} = 0$, $p(t \mid M_d) = \frac{cf_t}{cs}$

cf_t : raw count of term t in the collection

cs : raw collection size(total number of tokens in the collection)

Insufficient data

- Zero probabilities spell disaster
  - We need to smooth probabilities
    - Discount nonzero probabilities
    - Give some probability mass to unseen things
- There's a wide space of approaches to smoothing probability distributions to deal with this problem, such as adding 1, ½ or ε to counts, Dirichlet priors, discounting, and interpolation
  - [See FSNLP ch. 6 or CS224N if you want more]
- A simple idea that works well in practice is to use a mixture between the document multinomial and the collection multinomial distribution

Mixture model

- $P(w \mid d) = \lambda P_{mle}(w \mid M_d) + (1 - \lambda) P_{mle}(w \mid M_c)$
- Mixes the probability from the document with the general collection frequency of the word.
- Correctly setting λ is very important
- A high value of lambda makes the search "conjunctive-like" – suitable for short queries
- A low value is more suitable for long queries
- Can tune λ to optimize performance
  - Perhaps make it dependent on document size (cf. Dirichlet prior or Witten-Bell smoothing)

Basic mixture model summary

- General formulation of the LM for IR

$$p(Q,d) = p(d)\prod_{t\in Q}((1-\lambda)\,p(t)+\lambda p(t\mid M_d))$$

general language model

individual-document model

  - The user has a document in mind, and generates the query from this document.
  - The equation represents the probability that the document that the user had in mind was in fact this one.

Example

- Document collection (2 documents)
  - d_1: Xerox reports a profit but revenue is down
  - d_2: Lucent narrows quarter loss but revenue decreases further
- Model: MLE unigram from documents; $\lambda = ½$
- Query: *revenue down*
  - $P(Q \mid d_1) = [(1/8 + 2/16)/2] \times [(1/8 + 1/16)/2]$
    $= 1/8 \times 3/32 = 3/256$
  - $P(Q \mid d_2) = [(1/8 + 2/16)/2] \times [(0 + 1/16)/2]$
    $= 1/8 \times 1/32 = 1/256$
- Ranking: $d_1 > d_2$

Quiz: λ in the Mixture model

- Which of the following is *incorrect* about the purpose of λ in the mixture model:

a) λ balances the probability from the document with the probability from the whole collection.

b) A high value of λ is suitable for short queries.

c) The use of λ prevents zero probability disasters.

d) λ can be tuned to optimize performance.

Language models: pro & con

- Novel way of looking at the problem of text retrieval based on probabilistic language modeling
  - Conceptually simple and explanatory
  - Formal mathematical model
  - Natural use of collection statistics, not heuristics (almost…)
- LMs provide effective retrieval and can be improved to the extent that the following conditions can be met
  - Our language models are accurate representations of the data.
  - Users have some sense of term distribution.*
    - *Or we get more sophisticated with translation model

Comparison With Vector Space

- There's some relation to traditional tf.idf models:
  - (unscaled) term frequency is directly in model
  - the probabilities do length normalization of term frequencies
  - the effect of doing a mixture with overall collection frequencies is a little like idf: terms rare in the general collection but common in some documents will have a greater influence on the ranking

Comparison With Vector Space

- Similar in some ways
  - Term weights based on frequency
  - Terms often used as if they were independent
  - Inverse document/collection frequency used
  - Some form of length normalization useful

- Different in others
  - Based on probability rather than similarity
    - Intuitions are probabilistic rather than geometric
  - Details of use of document length and term, document, and collection frequency differ

LM vs. Prob. Model for IR

- The main difference is whether "Relevance" figures explicitly in the model or not
  - LM approach attempts to do away with modeling relevance
- LM approach asssumes that documents and expressions of information problems are of the same type
- Computationally tractable, intuitively appealing

LM vs. Prob. Model for IR

- Problems of basic LM approach
  - Assumption of equivalence between document and information need representation is unrealistic
  - Very simple models of language
  - Relevance feedback is difficult to integrate, as are user preferences, and other general issues of relevance
  - Can't easily accommodate phrases, passages, Boolean operators
- Current extensions focus on putting relevance back into the model, etc.

Resources

J.M. Ponte and W.B. Croft. 1998. A language modelling approach to information retrieval. In *SIGIR 21*.

D. Hiemstra. 1998. A linguistically motivated probabilistic model of information retrieval. *ECDL 2*, pp. 569–584.

A. Berger and J. Lafferty. 1999. Information retrieval as statistical translation. *SIGIR 22*, pp. 222–229.

D.R.H. Miller, T. Leek, and R.M. Schwartz. 1999. A hidden Markov model information retrieval system. *SIGIR 22*, pp. 214–221.

[Several relevant newer papers at *SIGIR 23–25,* 2000–2002.]

Workshop on Language Modeling and Information Retrieval, CMU 2001. http://la.lti.cs.cmu.edu/callan/Workshops/lmir01/ .

The Lemur Toolkit for Language Modeling and Information Retrieval. http://www-2.cs.cmu.edu/~lemur/ . CMU/Umass LM and IR system in C(++), currently actively developed.

Probabilistic relevance feedback

- Rather than reweighting in a vector space…
- If user has told us some relevant and some irrelevant documents, then we can proceed to build a probabilistic classifier, such as a Naive Bayes model:
  - $P(t_k|R) = |\mathbf{D}_{rk}| \;/\; |\mathbf{D}_r|$
  - $P(t_k|NR) = |\mathbf{D}_{nrk}| \;/\; |\mathbf{D}_{nr}|$
    - t_k is a term; $\mathbf{D}_r$ is the set of known relevant documents; $\mathbf{D}_{rk}$ is the subset that contain t_k; $\mathbf{D}_{nr}$ is the set of known irrelevant documents; $\mathbf{D}_{nrk}$ is the subset that contain t_k.

Probabilistic Relevance Feedback

1. Guess a preliminary probabilistic description of R and use it to retrieve a first set of documents V, as above.
2. Interact with the user to refine the description: learn some definite members of R and NR
3. Reestimate p_i and r_i on the basis of these
   - Or can combine new information with original guess (use Bayesian prior):

$$p_i^{(2)} = \frac{|V_i| + \kappa p_i^{(1)}}{|V| + \kappa}$$

κ is prior weight

4. Repeat, thus generating a succession of approximations to R.