Text Mining, part 2

Laura Brown

Some slides adapted from P. Smyth; Han, Kamber, & Pei;
Tan, Steinbach, & Kumar; C. Volinsky; R. Tibshirani; D. Kauchak
and http://nlp.stanford.edu/IR-book/

Text Mining: Main Types

- Retrieval – large corpus of text documents and I want the one closest to a specified query
  - e.g., web search, library catalogs, legal and medical precedent studies

- Analysis – have a bunch of text of interest, tell me something new about it
  - Classification and Clustering
  - Sentiment analysis, “buzz” searches

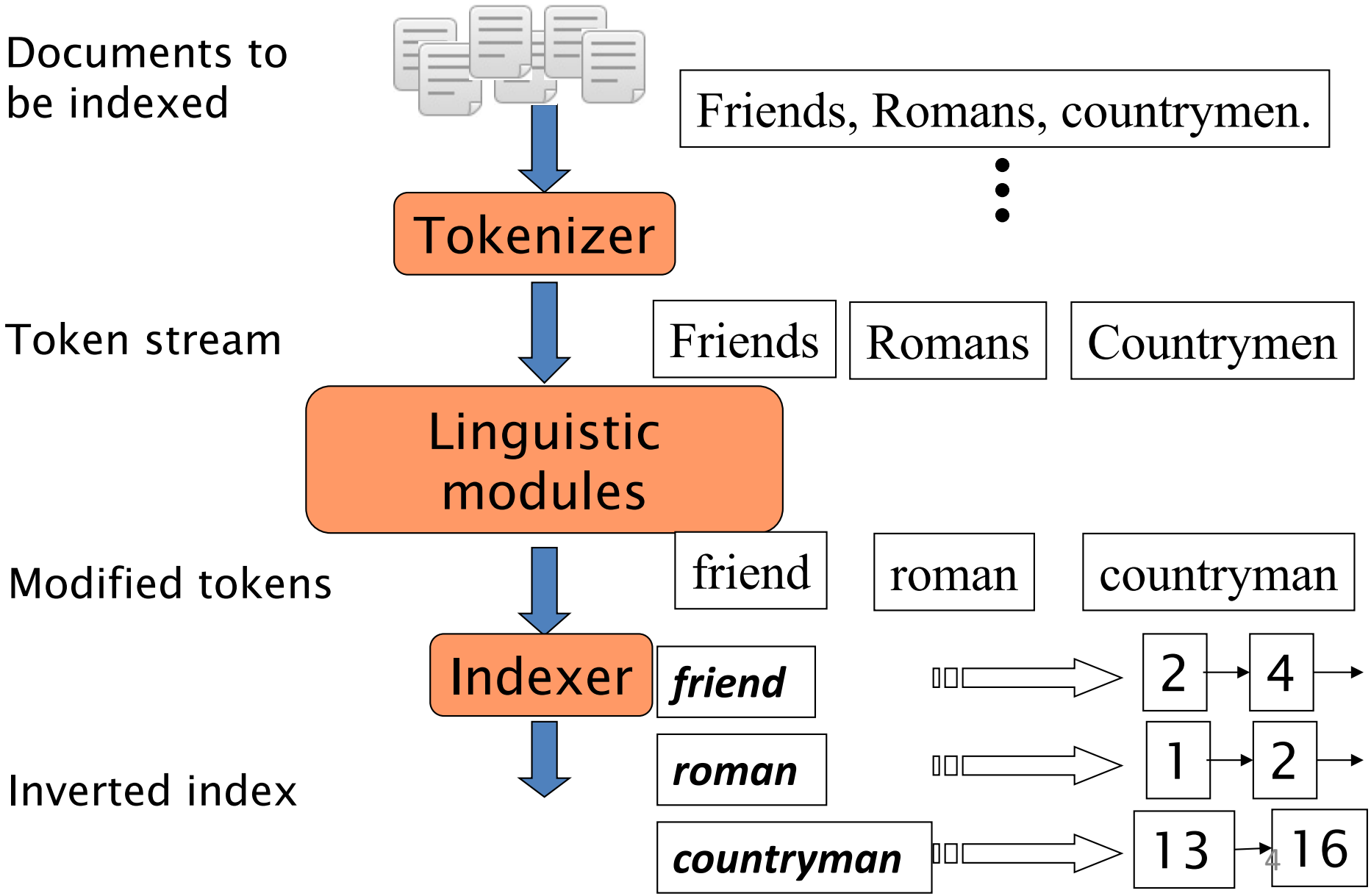
Outline

- Information Retrieval - *done*

- Text Classification
  - Text pre-processing
  - Naïve Bayes two models
  - Other classification methods

- Text Clustering
- Information Extraction

Recall the basic indexing pipeline



Parsing a document

- What format is it in?
  - pdf/word/excel/html?
- What language is it in?
- What character set is in use?
  - (CP1252, UTF-8, …)

Each of these is a classification problem, which we will study later in the course.

But these tasks are often done heuristically …

Complications: Format/language

- Documents being indexed can include docs from many different languages
  - A single index may contain terms from many languages.
- Sometimes a document or its components can contain multiple languages/formats
  - French email with a German pdf attachment.
  - French email quote clauses from an English-language contract

- There are commercial and open source libraries that can handle a lot of this stuff

Complications: What is a document?

We return from our query "documents" but there are often interesting questions of grain size:

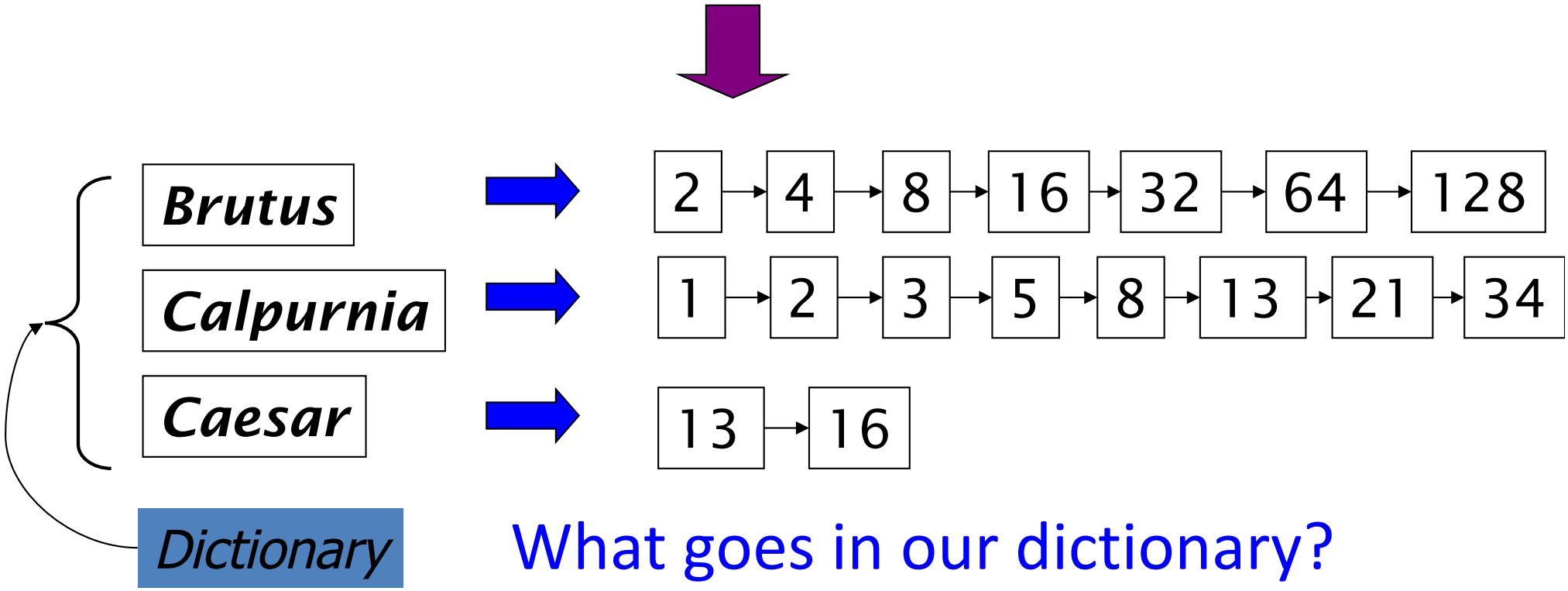
What is a unit document?

- A file?
- An email? (Perhaps one of many in a single mbox file)
  - What about an email with 5 attachments?
- A group of files (e.g., PPT or LaTeX split over HTML pages)

Text pre-processing

- Assume we've figured all of this out and we now have a stream of characters that is our document

"Friends, Romans, Countrymen …"



What goes in our dictionary?

Tokenization

- Input: "***Friends, Romans and Countrymen***"
- Output: Tokens
  - ***Friends***
  - ***Romans***
  - ***Countrymen***
- A token is an instance of a sequence of characters
- Each such token is now a candidate for an index entry, after further processing
  - Described below
- But what are valid tokens to emit?

Tokenization

- Issues in tokenization:
  - ***Finland's capital*** $\rightarrow$

    Finland AND ***s***? ***Finlands***? ***Finland's***?
  - ***Hewlett-Packard*** $\rightarrow$ ***Hewlett*** and ***Packard*** as two tokens?
    - ***state-of-the-art***: break up hyphenated sequence.
    - ***co-education***
    - ***lowercase***, ***lower-case***, ***lower case*** ?
    - It can be effective to get the user to put in possible hyphens
  - ***San Francisco***: one token or two?
    - How do you decide it is one token?

Numbers

- ***3/20/91 Mar. 12, 1991 20/3/91***
- ***55 B.C.***
- ***B-52***
- ***My PGP key is 324a3df234cb23e***
- ***(800) 234-2333***
  - Often have embedded spaces
  - Older IR systems may not index numbers
    - But often very useful: think about things like looking up error codes/stacktraces on the web
    - (One answer is using n-grams: IIR ch. 3)
  - Will often index “meta-data” separately
    - Creation date, format, etc.

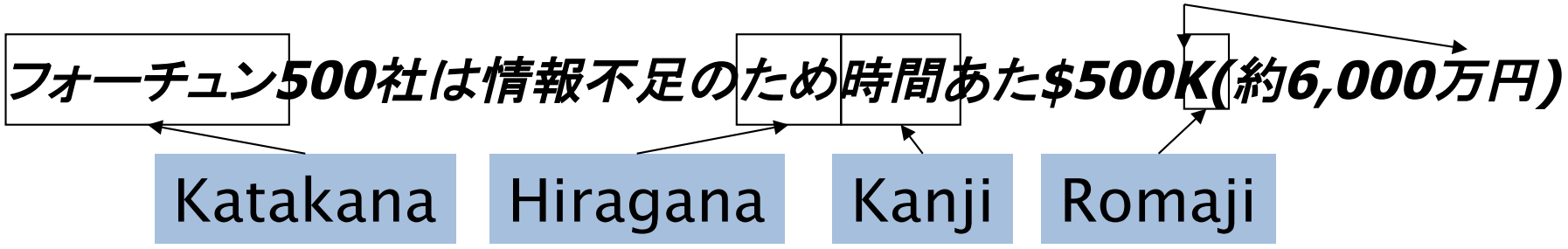
Tokenization: language issues

- French
  - ***L'ensemble*** → one token or two?
    - ***L*** ? ***L'*** ? ***Le*** ?
    - Want ***l'ensemble*** to match with ***un ensemble***
      - Until at least 2003, it didn't on Google
        - » Internationalization!

- German noun compounds are not segmented
  - ***Lebensversicherungsgesellschaftsangestellter***
  - 'life insurance company employee'
  - German retrieval systems benefit greatly from a **compound splitter** module
    - Can give a 15% performance boost for German

Tokenization: language issues

- Chinese and Japanese have no spaces between words:
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
  - Not always guaranteed a unique tokenization
- Further complicated in Japanese, with multiple alphabets intermingled
  - Dates/amounts in multiple formats



End-user can express query entirely in hiragana!

Tokenization: language issues

- Arabic (or Hebrew) is basically written right to left, but with certain items like numbers written left to right
- Words are separated, but letter forms within a word form complex ligatures

استقلت الجزائر في سنة 1962 بعد 132 عاما من الاحتلال الفرنسي.

- ← → ← → ← start
- 'Algeria achieved its independence in 1962 after 132 years of French occupation.'
- With Unicode, the surface presentation is complex, but the stored form is straightforward

Stop words

- With a stop list, you exclude from the dictionary entirely the commonest words. Intuition:
  - They have little semantic content: *the, a, and, to, be*
  - There are a lot of them: ~30% of postings for top 30 words
- But the trend is away from doing this:
  - Good compression techniques means the space for including stop words in a system is very small
  - Good query optimization techniques mean you pay little at query time for including stop words.
  - You need them for:
    - Phrase queries: “King of Denmark”
    - Various song titles, etc.: “Let it be”, “To be or not to be”
    - “Relational” queries: “flights to London”

Normalization to terms

- We may need to "normalize" words in indexed text as well as query words into the same form
  - We want to match ***U.S.A.*** and ***USA***
- Result is terms: a term is a (normalized) word type, which is an entry in our IR system dictionary
- We most commonly implicitly define equivalence classes of terms by, e.g.,
  - deleting periods to form a term
    - ***U.S.A., USA*** (***USA***
  - deleting hyphens to form a term
    - ***anti-discriminatory, antidiscriminatory*** (***antidiscriminatory***

Normalization: other languages

- Accents: e.g., French ***résumé*** vs. ***resume*.**
- Umlauts: e.g., German: ***Tuebingen*** vs. ***Tübingen***
  - Should be equivalent
- Most important criterion:
  - How are your users like to write their queries for these words?

- Even in languages that standardly have accents, users often may not type them
  - Often best to normalize to a de-accented term
    - ***Tuebingen, Tübingen, Tubingen*** ⎩ ***Tubingen***

Normalization: other languages

- Normalization of things like date forms
  - ***7月30日 vs. 7/30***
  - ***Japanese use of kana vs. Chinese characters***

- Tokenization and normalization may depend on the language and so is intertwined with language detection

Morgen will ich in MIT …

Is this German "mit"?

- Crucial: Need to "normalize" indexed text as well as query terms identically

Case folding

- Reduce all letters to lower case
  - exception: upper case in mid-sentence?
    - e.g., General Motors
    - Fed vs. fed
    - SAIL vs. sail
  - Often best to lower case everything, since users will use lowercase regardless of ‘correct’ capitalization…

- Longstanding Google example: [fixed in 2011…]
  - Query C.A.T.
  - #1 result is for “cats” (well, Lolcats) not Caterpillar Inc.

Normalization to terms

- An alternative to equivalence classing is to do asymmetric expansion
- An example of where this may be useful
  - Enter: ***window*** Search: ***window, windows***
  - Enter: ***windows*** Search: ***Windows, windows, window***
  - Enter: ***Windows*** Search: ***Windows***
- Potentially more powerful, but less efficient

Thesauri and soundex

- Do we handle synonyms and homonyms?
  - E.g., by hand-constructed equivalence classes
    - ***car*** = ***automobile*** ***color*** = ***colour***
  - We can rewrite to form equivalence-class terms
    - When the document contains ***automobile***, index it under ***car-automobile*** (and vice-versa)
  - Or we can expand a query
    - When the query contains ***automobile***, look under ***car*** as well
- What about spelling mistakes?
  - One approach is Soundex, which forms equivalence classes of words based on phonetic heuristics

Lemmatization

- Reduce inflectional/variant forms to base form
- E.g.,
  - *am, are, is* → *be*
  - *car, cars, car's, cars'* → *car*
- *the boy's cars are different colors* → *the boy car be different color*
- Lemmatization implies doing "proper" reduction to dictionary headword form

Stemming

- Reduce terms to their "roots" before indexing
- "Stemming" suggests crude affix chopping
  - language dependent
  - e.g., ***automate(s), automatic, automation*** all reduced to ***automat***.

for example compressed and compression are both accepted as equivalent to compress.

for exampl compress and compress ar both accept as equival to compress

Porter's algorithm

- Most common algorithm for stemming English
  - Results suggest it's at least as good as other stemming options
- Conventions + 5 phases of reductions
  - phases applied sequentially
  - each phase consists of a set of commands
  - sample convention: *Of the rules in a compound command, select the one that applies to the longest suffix.*

Typical rules in Porter

- *sses* → *ss*
- *ies* → *i*
- *ational* → *ate*
- *tional* → *tion*

- Weight of word sensitive rules
- *(m>1) EMENT* →
  - *replacement* → *replac*
  - *cement* → *cement*

Other stemmers

- Other stemmers exist:
  - Lovins stemmer
    - http://www.comp.lancs.ac.uk/computing/research/stemming/general/lovins.htm
    - Single-pass, longest suffix removal (about 250 rules)
  - Paice/Husk stemmer
  - Snowball
- Full morphological analysis (lemmatization)
  - At most modest benefits for retrieval

Language-specificity

- The above methods embody transformations that are
  - Language-specific, and often
  - Application-specific
- These are “plug-in” addenda to the indexing process
- Both open source and commercial plug-ins are available for handling these

Does stemming help?

- English: very mixed results. Helps recall for some queries but harms precision on others
  - E.g., operative (dentistry) ⇒ oper
- Definitely useful for Spanish, German, Finnish, …
  - 30% performance gains for Finnish!

What normalization techniques to use…

- What is the size of the corpus?
  - small corpora often require more normalization
- Depends on the users and the queries
- Query suggestion (i.e. “did you mean”) can often be used instead of normalization
- Most major search engines do little to normalize data except lowercasing and removing punctuation (and not even these always)

Outline

- Information Retrieval - *done*
- Text Classification
  - Text pre-processing
  - Naïve Bayes two models
  - Other classification methods
- Text Clustering
- Information Extraction

Problems in Text Classification

- Is this spam?

to Recipients

Be careful with this message. Many people marked similar messages as phishing scams, so this might contain unsafe content. Learn more

Good Day,

We are private investor and we give out Guarantee Business Loans, Automobile Purchase Loans,House Purchase Loans and other Personal Loans E.T.C maximum with 4% .contact us: xloaninvestment@gmail.com

Kindly visit our website on www.xloan.info

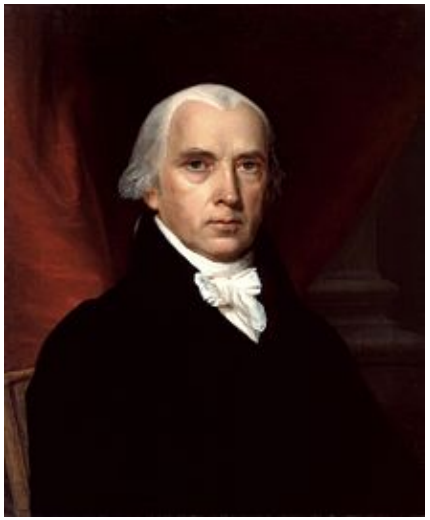
Just fill this little form Below
Name:
Loan Amount:
Expected Repayment Duration:
Phone no:

Email : xloaninvestment@gmail.com

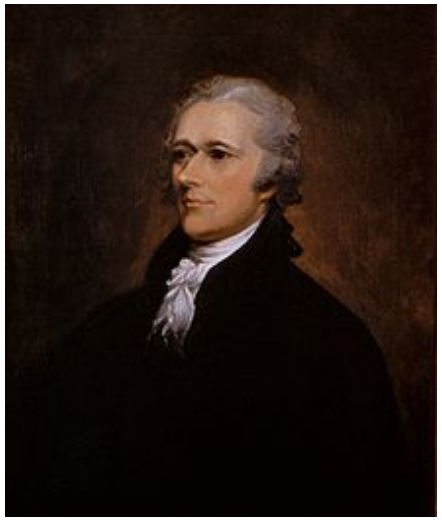
Regards
Mr H. Perry
Manager

Authorship Identification

- Who wrote the Federalist papers?
  - 1787-1788: anonymous essays try to convince New York to ratify US constitution: Jay Madison, Hamilton
  - Authorship of 12 letters in dispute
  - 1963: solved by Mostellar and Wallace using Bayesian methods



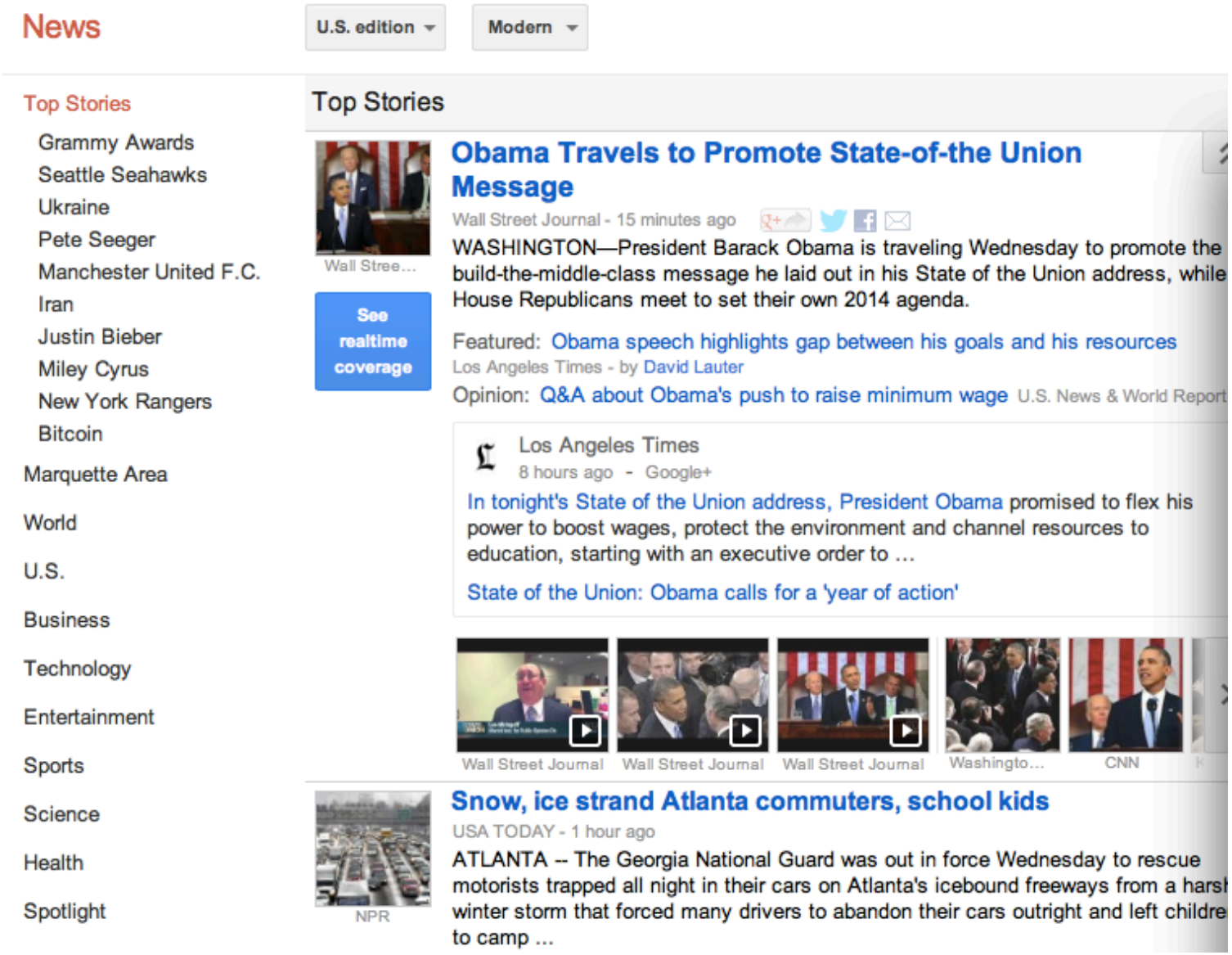
James Madison



Alexander Hamilton

John Jay

News Categories



Sentiment Analysis

- very much overrated
- I think a brilliant performance by Hopkins is utterly wasted in this scurrilous piece of trash.
- Inaccurate twaddle.
- A very special kinda movie for a very special kinda person.. nuff said.
- Good movie for kids, plenty of excitement...keeps children attentive.
- Although cheesy, it's a must-see movie, especially during Christmas.
- absolutely amazing movie..the special effects were out of this world..keanu reeves acted brilliantly and was supported really well by lawrence

Article Subject Identification

Brain and Cognition

Syntactic frame and verb bias in aphasia: Plausibility judgments of undergoer-subject sentences

- MESH Subject Category
  - Chemistry
  - Drug Therapy
  - Embryology
  - Epidemiology
  - …

Other Text Classification Problem

- Assign subject categories, topics, or genres
- Authorship Age/gender identification
- Language identification
- Sentiment Analysis

Text Classification Methods - Manual

- Used by Yahoo! (originally), Looksmart, about.com, Open Directory Project (http://www.dmoz.org), PubMed
- Very accurate; job is done by experts
- Consistent when the problem size and team is small
- Difficult and expensive to scale

Classification Methods - Automatic

- Hand Coded Rule-based Systems
  - Common for span filters
  - Companies (e.g., Verity) provide “IDE” for writing rules
    - For example, assign category if document contains a given Boolean combination of words
  - Commercial systems have standing rules – can have very high accuracy if rule has been refined over time by a subject expert
  - Building and maintaining rules is expensive

Classification Methods – Supervised Learning

- Many systems rely partly on machine learning (Autonomy, MSN, Verity, Enkata, Yahoo!, …)
  - Naïve Bayes (simple, common method)
  - Knn (simple, powerful)
  - Support Vector Machines (newer)
  - …
  - Requires hand-labeling training data
- Most commercial systems use a mixture of methods

Naïve Bayes for Text Classification

- Given: Classify a new document D with a tuple of attribute values $D = (x_1, x_2, \ldots, x_p)$ into one of the classes c_j in C

$$P(C \mid D) = \frac{P(D \mid C)P(C)}{P(D)}$$

$$c_{MAP} = \arg\max_{c_j \in C} \; P(x_1, x_2, \ldots, x_p \mid c_j)P(c_j)$$

Bag-of-Words Representation

- Bag-of-words representation
  - List all the words and how many times they appear, or
  - List all the words and if they appear
- Easy to generate and use

Leonardo da Vinci

From Wikipedia, the free encyclopedia

"Da Vinci" redirects here. For other uses, see Da Vinci (disambiguation).

Leonardo di ser Piero da Vinci (Italian pronunciation: [leoˈnardo da ˈvintʃi] pronunciation (help·info)) (April 15, 1452 – May 2, 1519, Old Style) was an Italian Renaissance polymath: painter, sculptor, architect, musician, mathematician, engineer, inventor, anatomist, geologist, cartographer, botanist, and writer. His genius, perhaps more than that of any other figure, epitomized the Renaissance humanist ideal. Leonardo has often been described as the archetype of the Renaissance Man, a man of "unquenchable curiosity" and "feverishly inventive imagination".[1] He is widely considered to be one of the greatest painters of all time and perhaps the most diversely talented person ever to have lived.[2] According to art historian Helen Gardner, the scope and depth of his interests were without precedent and "his mind and personality seem to us superhuman, the man himself mysterious and remote".[1] Marco Rosci states that while there is much speculation about Leonardo, his vision of the world is essentially logical rather than mysterious, and that the empirical methods he employed were unusual for his time.[3]

Born out of wedlock to a notary, Piero da Vinci, and a peasant woman, Caterina, at Vinci in the region of Florence, Leonardo was educated in the studio of the renowned Florentine painter, Verrocchio. Much of his earlier working life was spent in the service of Ludovico il Moro in

Bag of Words Representation

- Bag-of-words representation of a document
  - List all the words, how many times they appear
  - `painter=26, Renaissance=34; scientific=10; figure=12`
- Very easy to generate and easy to use, but is it too much of a reduction?
  - Idea: by itself "figure" can take on multiple meanings, but we can learn from other words in the document
    - `Landscape, composition, silhouette`, suggest figure in a work of art
    - `Calculate, evaluate, math`, suggest an alternative meaning of figure

Counting Words

- Make a list of all words present in the documents
- Index the words, $w = 1, \ldots, W$ (e.g., alphabetical order) and index the documents $d = 1, \ldots, D$
- For each document d, count how many times each word w appears let this be x_{dw}
  The dth document is $(x_{d1}, x_{d2}, \ldots, x_{dW})$
- The test documents should be constructed the same way $d_{test} = (x_{test1}, x_{test2}, \ldots, x_{testW})$
- Length of the feature vector is the vocabulary, $|W| = V$

Simple Example

- Document, Class:
  1. ”Laura likes math”, +
  2. “Andy hates, hates, math”, -
- Test Sample: “hates math”

The example has *D=2* documents and *W*=5 words. The counts for each document are:

	Andy	hates	Laura	like	math	Class
d_1	0	0	1	1	1	+
d_2	1	2	0	0	1	-
d_{test}	0	1	0	0	1	?

This is the document-term (or term-doc, TD) matrix

Multinomial Model for NB

- The probability of a document *d* being in class *c* is computed as

$$P(c \mid d) \propto P(c) \prod_{1 \le k \le V} P(x_k \mid c)$$

- The best class is selected as:

$$c_{MAP} = \arg\max_{c_j \in C} \hat{P}(c_j) \prod_{1 \le k \le V} \hat{P}(x_k \mid c_j)$$

$$c_{MAP} = \arg\max_{c_j \in C} [\log \hat{P}(c_j) + \sum_{1 \le k \le V} \log \hat{P}(x_k \mid c_j)]$$

Multinomial Model for NB

- Estimate probabilities

$$\hat{P}(c_j) = \frac{\#(C = c_j)}{D}$$

$$\hat{P}(x_i \mid c_j) = \frac{T_{jx}}{\sum_{x' \in V} T_{jx'}}$$

- Smoothed estimates:

$$\hat{P}(x_i \mid c_j) = \frac{T_{jx} + 1}{(\sum_{x' \in V} T_{jx'}) + V}$$

Multinomial Model

- Representation: one feature x_i for each position in document
  - Feature's values are all words in the dictionary
- x_i = "word" at position *I*
- Naïve Bayes assumption
  - Given the document's topic, word in the document tells us nothing about words in other positions
  - *Positional independence assumption:* word appearance does not depend on position
  - Assume text is determine by generating x_1 from multinomial distribution over words, and similarly for x_2, x_3, …

Term-Doc incidence matrix

- The bag-of-words representation can also reflect 0/1 on whether a word appears in a document (rather than the frequency).
- Using the same example the matrix becomes:

	Andy	hates	Laura	like	math	Class
d_1	0	0	1	1	1	+
d_2	1	1	0	0	1	-
d_{test}	0	1	0	0	1	?

Bernoulli Model

- Follows classical Naïve Bayes approach

$$c_{MAP} = \arg\max_{c_j \in C} \; P(x_1, x_2, \ldots, x_p \mid c_j) P(c_j)$$

- Maximum likelihood estimates:

$$\hat{P}(c_j) = \frac{\#(C = c_j)}{D} \qquad \hat{P}(x_i \mid c_j) = \frac{\#(X_i = 1, C = c_j)}{\#(C = c_j)}$$

Bernoulli Model

- Representation: one feature x_w for each word in dictionary
- x_{dw} = *true* if word *w* appears in document *d*
- Naïve Bayes assumption
  - Given the document's topic, appearance of one word in the document tells us nothing about chances that another word appears
- Assumes text is generated by running through the dictionary and independently determining whether to include word *i* based on probabilities

NB Bayes Model

- Both the Bernoulli and Multinomial NB models are described in detail in the Information Retrieval book linked from Canvas.

- The book has methods to train a model and test documents using both models; you should follow this in Project 4!

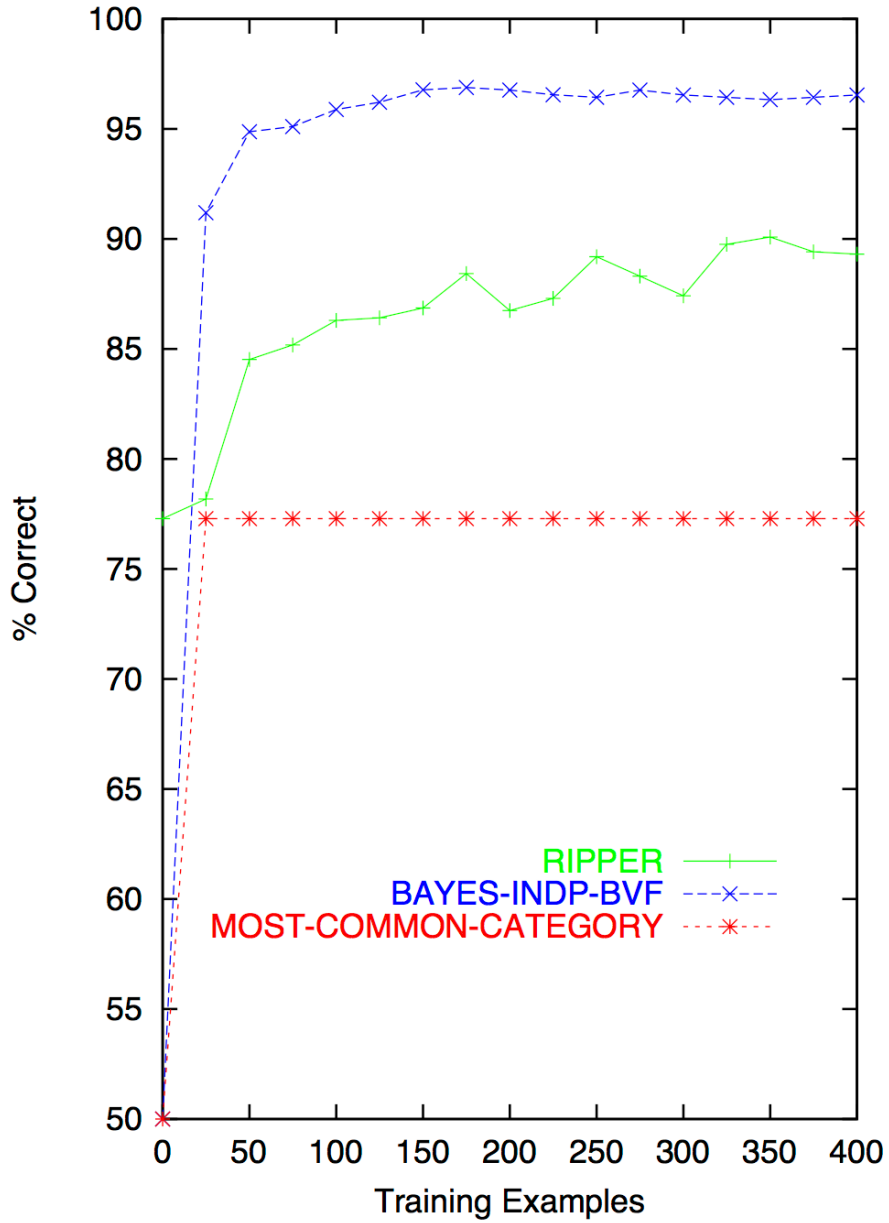
Application of NB

WebKB Experiment (1998)

- Classify webpages from CS departments into:
  - Student, faculty, course, project
- Train on ~5,000 hand-labeled web pages
- Crawl and classify a new site (CMU)
- 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%

NB on Spam


100
95
90
85
80
75
70
65
60
55
50
% Correct
0
50
100
150
200
250
300
350
400
Training Examples
RIPPER
BAYES-INDP-BVF
MOST-COMMON-CATEGORY

Outline

- Information Retrieval - *done*
- Text Classification
  - Text pre-processing
  - Naïve Bayes two models
  - Other classification methods
- Text Clustering
- Information Extraction

Classification Methods : 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

- 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

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 commonest 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

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

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, 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

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
- 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

Other Classifiers for Text Mining

- K nearest neighbor
  - 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 some cases it's more accurate than NB
  - KNN has high variance and low bias
  - NB has low variance and high bias

Text Classification

- Naive Bayes classifier
  - Simple, cheap, high bias, linear
- K Nearest Neighbor classification
  - Simple, expensive at test time, high variance, non-linear
- Vector space classification: Rocchio
  - Simple linear discriminant classifier; perhaps too simple
- Support Vector Machines (SVMs)
  - Soft margin SVMs and kernels for non-linear classifiers
- Some empirical evaluation and comparison
- Text-specific issues in classification

Example of Text Categorization

- Classic Reuters-21578 Data Set
- Most (over)used data set
- 21578 documents
- 9603 training, 3299 test articles (ModApte/Lewis split)
- 118 categories
  - An article can be in more than one category
  - Learn 118 binary category distinctions
- Average document: about 90 types, 200 tokens
- Average number of classes assigned
  - 1.24 for docs with at least one category
- Only about 10 out of 118 categories are large

Common categories
(#train, #test)

- Earn (2877, 1087)
- Acquisitions (1650, 179)
- Money-fx (538, 179)
- Grain (433, 149)
- Crude (389, 189)
- Trade (369,119)
- Interest (347, 131)
- Ship (197, 89)
- Wheat (212, 71)
- Corn (182, 56)

Reuters Data Set

<REUTERS TOPICS="YES" LEWISSPLIT="TRAIN" CGISPLIT="TRAINING-SET" OLDID="12981" NEWID="798">

<DATE> 2-MAR-1987 16:51:43.42</DATE>

<TOPICS><D>livestock</D><D>hog</D></TOPICS>

<TITLE>AMERICAN PORK CONGRESS KICKS OFF TOMORROW</TITLE>

<DATELINE> CHICAGO, March 2 - </DATELINE><BODY>The American Pork Congress kicks off tomorrow, March 3, in Indianapolis with 160 of the nations pork producers from 44 member states determining industry positions on a number of issues, according to the National Pork Producers Council, NPPC.

Delegates to the three day Congress will be considering 26 resolutions concerning various issues, including the future direction of farm policy and the tax law as it applies to the agriculture sector. The delegates will also debate whether to endorse concepts of a national PRV (pseudorabies virus) control and eradication program, the NPPC said.

A large trade show, in conjunction with the congress, will feature the latest in technology in all areas of the industry, the NPPC added. Reuter

&#3;</BODY></TEXT></REUTERS>

Per class evaluation measures

- Recall: Fraction of docs in class *i* classified correctly:

$$\frac{c_{ii}}{\sum_j c_{ij}}$$

- Precision: Fraction of docs assigned class *i* that are actually about class *i*:

$$\frac{c_{ii}}{\sum_j c_{ji}}$$

- Accuracy: (1 - error rate) Fraction of docs classified correctly:

$$\frac{\sum_i c_{ii}}{\sum_j \sum_i c_{ij}}$$

Micro- vs. Macro-Averaging

- If we have more than one class, how do we combine multiple performance measures into one quantity?
- Macroaveraging: Compute performance for each class, then average.
- Microaveraging: Collect decisions for all classes, compute contingency table, evaluate.

Micro- vs. Macro-Averaging: Example

Class 1

	Truth: yes	Truth: no
Classifier: yes	10	10
Classifier: no	10	970

Class 2

	Truth: yes	Truth: no
Classifier: yes	90	10
Classifier: no	10	890

Micro Ave. Table

	Truth: yes	Truth: no
Classifier: yes	100	20
Classifier: no	20	1860

- Macroaveraged precision: (0.5 + 0.9)/2 = 0.7
- Microaveraged precision: 100/120 = .83

- Microaveraged score is dominated by score on common classes

(a)

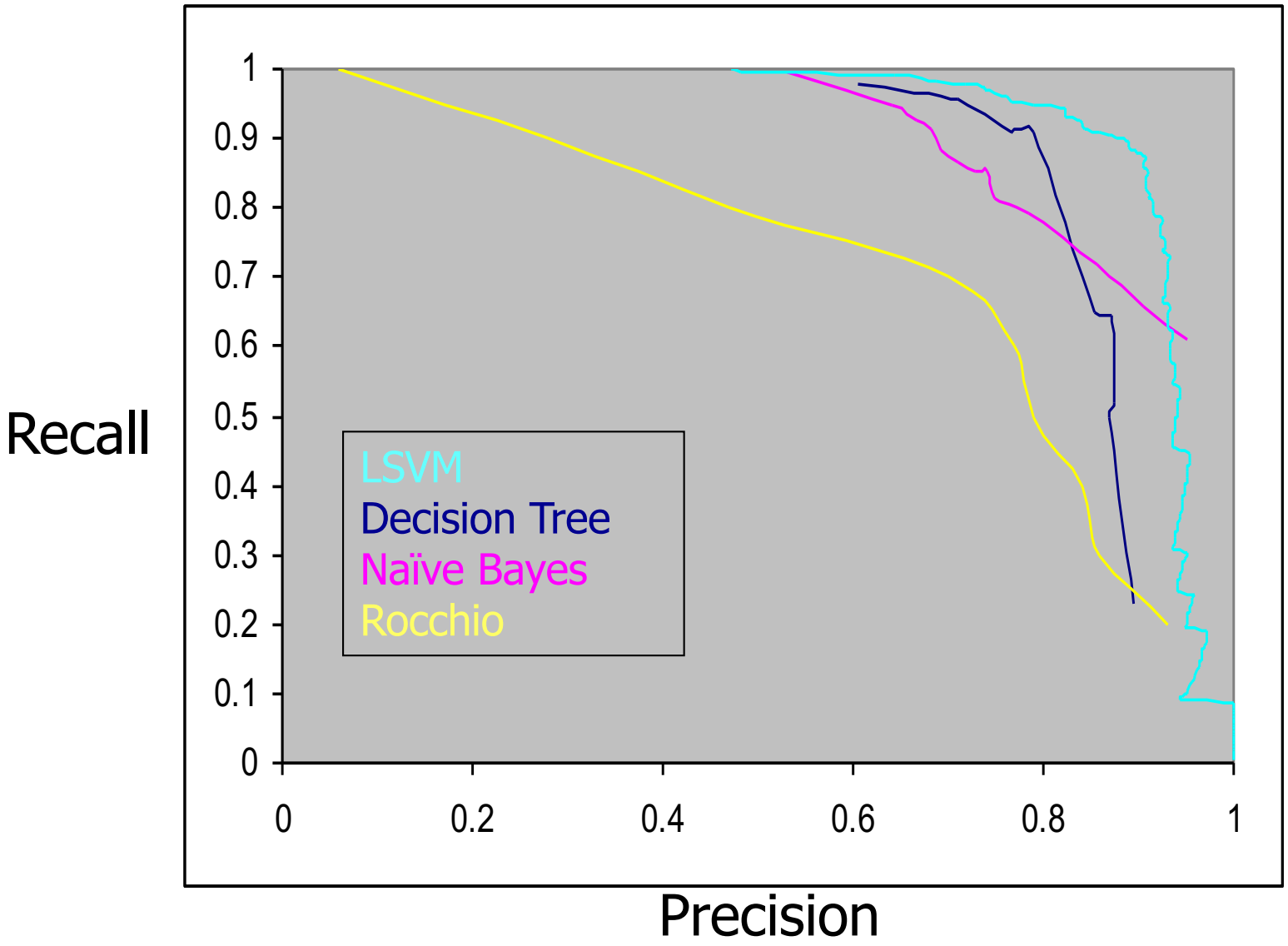
	NB	Rocchio	kNN	SVM
micro-avg-L (90 classes)	80	85	86	89
macro-avg (90 classes)	47	59	60	60

(b)

	NB	Rocchio	kNN	trees	SVM
earn	96	93	97	98	98
acq	88	65	92	90	94
money-fx	57	47	78	66	75
grain	79	68	82	85	95
crude	80	70	86	85	89
trade	64	65	77	73	76
interest	65	63	74	67	78
ship	85	49	79	74	86
wheat	70	69	77	93	92
corn	65	48	78	92	90
micro-avg (top 10)	82	65	82	88	92
micro-avg-D (118 classes)	75	62	n/a	n/a	87

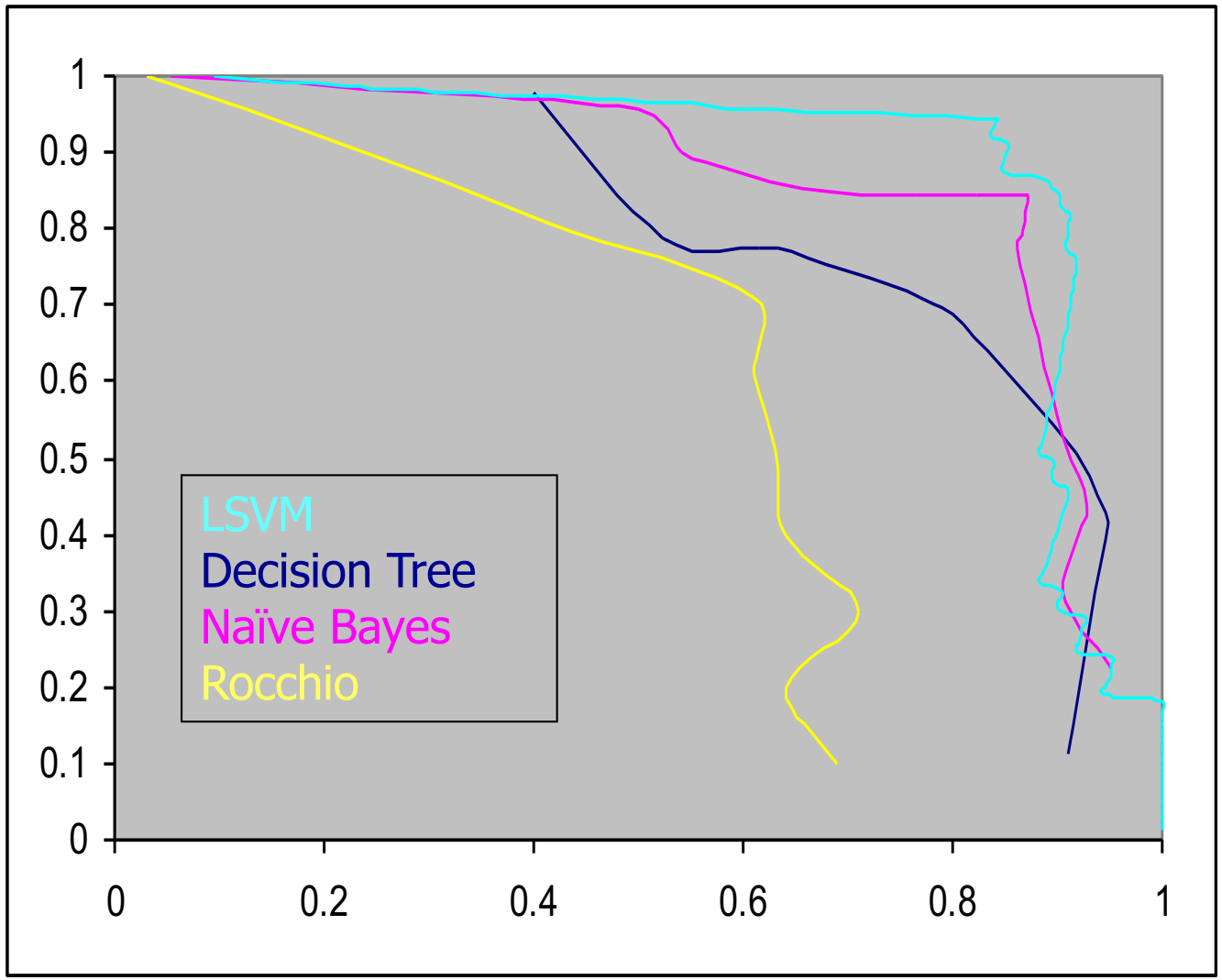
Evaluation measure: F_1

Precision-recall for category: Crude



Dumais (1998)

Precision-recall for category: Ship



Dumais (1998)

The Real World

- Gee, I'm building a text classifier for real, now!
- What should I do?

- How much training data do you have?
  - None
  - Very little
  - Quite a lot
  - A huge amount and its growing

Manually written rules

- No training data, adequate editorial staff?
- Never forget the hand-written rules solution!
  - If (wheat or grain) and not (whole or bread) then
    - Categorize as grain
- In practice, rules get a lot bigger than this
  - Can also be phrased using tf or tf.idf weights
- With careful crafting (human tuning on development data) performance is high:
  - Construe: 94% recall, 84% precision over 675 categories (Hayes and Weinstein 1990)
- Amount of work required is huge
  - Estimate 2 days per class … plus maintenance

Very little data?

- If you're just doing supervised classification, you should stick to something high bias
  - There are theoretical results that Naïve Bayes should do well in such circumstances (Ng and Jordan 2002 NIPS)
- The interesting theoretical answer is to explore semi-supervised training methods:
  - Bootstrapping, EM over unlabeled documents, …
- The practical answer is to get more labeled data as soon as you can
  - How can you insert yourself into a process where humans will be willing to label data for you??

A reasonable amount of data?

- Perfect!
- We can use all our clever classifiers
- Roll out the SVM!

- But if you are using an SVM/NB etc., you should probably be prepared with the "hybrid" solution where there is a Boolean overlay
  - Or else to use user-interpretable Boolean-like models like decision trees
  - Users like to hack, and management likes to be able to implement quick fixes immediately

A huge amount of data?

- This is great in theory for doing accurate classification…
- But it could easily mean that expensive methods like SVMs (train time) or kNN (test time) are quite impractical

- Naïve Bayes can come back into its own again!
  - Or other advanced methods with linear training/test complexity like regularized logistic regression (though much more expensive to train)

How many categories?

- A few (well separated ones)?
  - Easy!
- A zillion closely related ones?
  - Think: Yahoo! Directory, Library of Congress classification, legal applications
  - Quickly gets difficult!
    - Classifier combination is always a useful technique
      - Voting, bagging, or boosting multiple classifiers
    - Much literature on hierarchical classification
      - Mileage fairly unclear, but helps a bit (Tie-Yan Liu et al. 2005)
      - Definitely helps for scalability, even if not in accuracy
    - May need a hybrid automatic/manual solution

How can one tweak performance?

- Aim to exploit any domain-specific useful features that give special meanings or that zone the data
  - E.g., an author byline or mail headers
- Aim to collapse things that would be treated as different but shouldn't be.
  - E.g., part numbers, chemical formulas

- Does putting in "hacks" help?
  - You bet!
    - Feature design and non-linear weighting is *very* important in the performance of real-world systems

The Real World

P. Jackson and I. Moulinier. 2002. *Natural Language Processing for Online Applications*

- "There is no question concerning the commercial value of being able to classify documents automatically by content. There are myriad potential applications of such a capability for corporate intranets, government departments, and Internet publishers"

- "Understanding the data is one of the keys to successful categorization, yet this is an area in which most categorization tool vendors are extremely weak. Many of the 'one size fits all' tools on the market have not been tested on a wide range of content types."