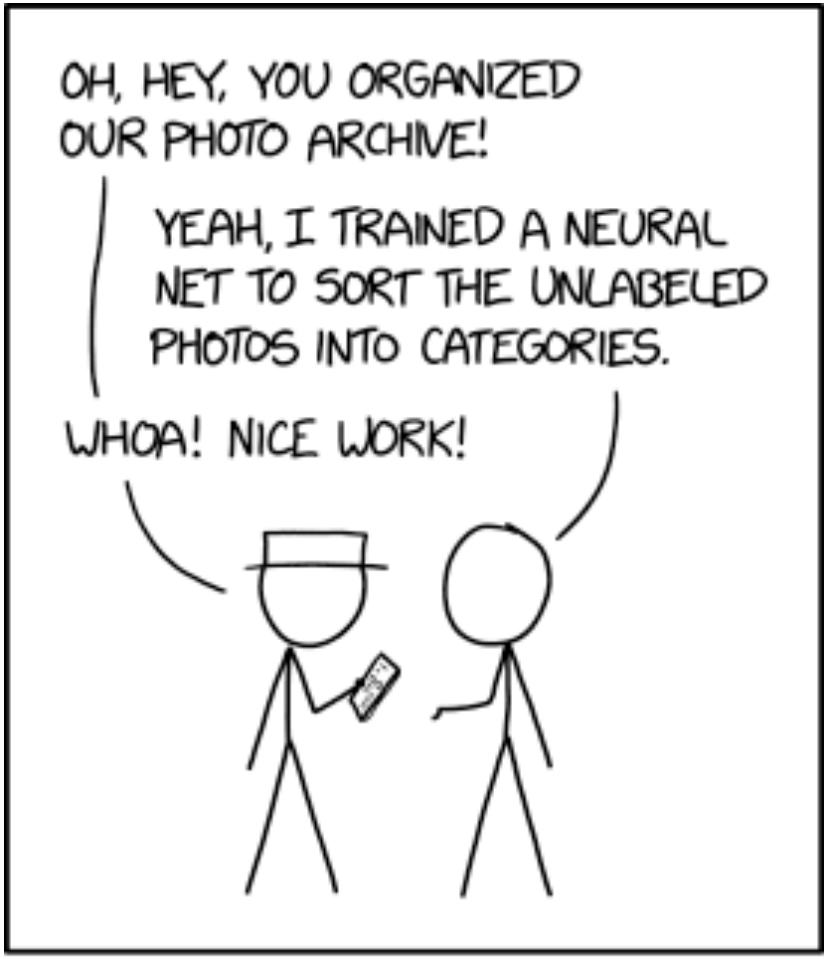



ENGINEERING TIP:
WHEN YOU DO A TASK BY HAND, YOU CAN TECHNICALLY SAY YOU TRAINED A NEURAL NET TO DO IT.

xkcd.com/2173/

# Text Mining and Information Retrieval: Part III

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

# Text Classification

# 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

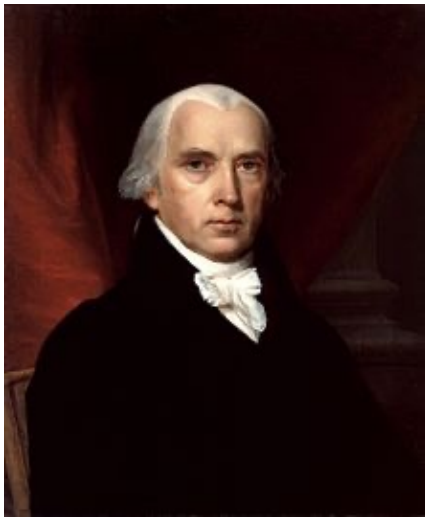
Just fill this little form Below
Name:
Loan Amount:
Expected Repayment Duration:
Phone no:

Email : xloaninvestment@gmail.com

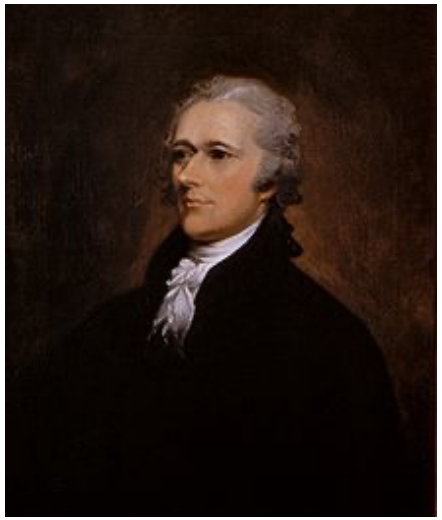
Regards
Mr H. Perry
Manager

# 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

# 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

# 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

# Classification Methods

- 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 - Automatic

- Hand-coded rule-based classifiers
  - Common for spam filters
  - One technique used by news agencies, intelligence agencies, etc.
  - Widely deployed in government and enterprise
  - Vendors provide “IDE” for writing such rules
- 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

- Supervised learning
  - Naive Bayes (simple, common)
  - k-Nearest Neighbors (simple, powerful)
  - Support-vector machines (newer, generally more powerful)
  - Decision trees → random forests → gradient-boosted decision trees (e.g., xgboost)
  - … plus many other methods
  - No free lunch: need hand-classified training data
  - But data can be built up by amateurs
- Many commercial systems use a mix of methods

# Text Classification: Naïve Bayes

# 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* in *C*

$$P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)}$$

$$\hat{c} = \arg\max_{c \in C} \; P(x_1, x_2, \ldots, x_p \mid c)P(c)$$

$$\hat{c} = \arg\max_{c \in C} \; P(c) \prod_{1 \leq i \leq p} P(x_i \mid c)$$

# Aside: Language models for IR

- Language modeling approach to IR
  - Key idea: a document is a good match to a query if the document model is likely to generate the query
    - This happens if the document contains the query words often
  - Build a probabilistic language model $M_d$ for each document *d* and ranks documents based on the probability of the model generating the query: $P(q \mid M_d)$
- A language model is a function that puts a probability measure over strings drawn from a vocabulary
  - The probability of a string/document can typically be found using chain rule:
    - $P(t_1 t_2 t_3 t_4)$

# Language Models

- Unigram Language Model – estimates each term independently:

$$P_{\text{uni}}(t_1 t_2 t_3 t_4) = P(t_1)P(t_2)P(t_3)P(t_4)$$

- Bigram Language Model – conditions on the previous term:

$$P_{\text{bi}}(t_1 t_2 t_3 t_4) = P(t_1)P(t_2 \mid t_1)P(t_3 \mid t_2)P(t_4 \mid t_3)$$

# Multinomial Model for NB

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

$$P(c \mid d) \propto P(c) \prod_{1 \le k \le n_d} P(t_k|c)$$

$t_k$ - token at position *k*

$n_d$ - number of tokens in *d*

- The best class is selected as:

$$\hat{c} = \arg\max_{c \in C} \hat{P}(c) \prod_{1 \le k \le n_d} \hat{P}(t_k \mid c)$$

$$\hat{c} = \arg\max_{c \in C} [\log \hat{P}(c) + \sum_{1 \le k \le n_d} \log \hat{P}(t_k \mid c)]$$

# Multinomial Model for NB

- Estimate probabilities

$$\hat{P}(c) = \frac{N_c}{N} \qquad \hat{P}(t \mid c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}}$$

$T_{ct}$ = num. of occurrences of term *t* in training docs of class *c*

- Smoothed estimates (add-one, Laplace)

$$\hat{P}(t \mid c) = \frac{T_{ct} + 1}{\sum_{t' \in V}(T_{ct'} + 1)} \text{ or } \frac{T_{ct} + 1}{(\sum_{t' \in V} T_{ct'}) + |V|}$$

*V* – Vocabulary, # of unique words in training data

# Methods for MultinomialNB, Fig. 13.2 IR book

TRAINMULTINOMIALNB($\mathbb{C}, \mathbb{D}$)

1 $V \leftarrow$ EXTRACTVOCABULARY($\mathbb{D}$)
2 $N \leftarrow$ COUNTDOCS($\mathbb{D}$)
3 **for each** $c \in \mathbb{C}$
4 **do** $N_c \leftarrow$ COUNTDOCSINCLASS($\mathbb{D}, c$)
5 $prior[c] \leftarrow N_c / N$
6 $text_c \leftarrow$ CONCATENATETEXTOFALLDOCSINCLASS($\mathbb{D}, c$)
7 **for each** $t \in V$
8 **do** $T_{ct} \leftarrow$ COUNTTOKENSOFTERM($text_c, t$)
9 **for each** $t \in V$
10 **do** $condprob[t][c] \leftarrow \frac{T_{ct}+1}{\sum_{t'}(T_{ct'}+1)}$
11 **return** $V, prior, condprob$

APPLYMULTINOMIALNB($\mathbb{C}, V, prior, condprob, d$)

1 $W \leftarrow$ EXTRACTTOKENSFROMDOC($V, d$)
2 **for each** $c \in \mathbb{C}$
3 **do** $score[c] \leftarrow \log prior[c]$
4 **for each** $t \in W$
5 **do** $score[c] \mathrel{+}= \log condprob[t][c]$
6 **return** $\arg\max_{c \in \mathbb{C}} score[c]$

# Bernoulli Model

- Follows classical Naïve Bayes approach, but relies on a multivariate Bernoulli model or a Bernoulli model
  - Indicator for each term of the vocabulary being in the document or not

- Probability estimates

$$\hat{P}(c) = \frac{N_c}{N} \qquad \hat{P}(t \mid c) = \frac{N_{ct} + 1}{N_c + 2}$$

# Methods for BernoulliNB, Fig. 13.3 IR book

TRAINBERNOULLINB($\mathbb{C}, \mathbb{D}$)
1 $V \leftarrow$ EXTRACTVOCABULARY($\mathbb{D}$)
2 $N \leftarrow$ COUNTDOCS($\mathbb{D}$)
3 **for each** $c \in \mathbb{C}$
4 **do** $N_c \leftarrow$ COUNTDOCSINCLASS($\mathbb{D}, c$)
5 $prior[c] \leftarrow N_c/N$
6 **for each** $t \in V$
7 **do** $N_{ct} \leftarrow$ COUNTDOCSINCLASSCONTAININGTERM($\mathbb{D}, c, t$)
8 $condprob[t][c] \leftarrow (N_{ct}+1)/(N_c+2)$
9 **return** $V, prior, condprob$

APPLYBERNOULLINB($\mathbb{C}, V, prior, condprob, d$)
1 $V_d \leftarrow$ EXTRACTTERMSFROMDOC($V, d$)
2 **for each** $c \in \mathbb{C}$
3 **do** $score[c] \leftarrow \log prior[c]$
4 **for each** $t \in V$
5 **do if** $t \in V_d$
6 **then** $score[c] \mathrel{+}= \log condprob[t][c]$
7 **else** $score[c] \mathrel{+}= \log(1 - condprob[t][c])$
8 **return** $\arg\max_{c \in \mathbb{C}} score[c]$

# Multinomial vs Bernoulli

- General Formula: $\hat{c} = \arg\max_{c \in C} P(d \mid c) P(c)$

$$\text{Multinomial} \quad P(d|c) = P(< t_1, \ldots, t_k, \ldots, t_{n_d} > \mid c)$$

$$\text{Bernoulli} \quad P(d|c) = P(< e_1, \ldots, e_i, \ldots, e_M > \mid c)$$

- Conditional Independence

$$\text{Multinomial} \quad P(d|c) = \prod_{1 \leq k \leq n_d} P(X_k = t_k \mid c)$$

$$\text{Bernoulli} \quad P(d|c) = \prod_{1 \leq i \leq M} P(U_i = e_i \mid c)$$

# NB Models

- Both the Bernoulli and Multinomial NB models are described in detail in the Information Retrieval book linked from Canvas. (Chapter 13)

- How can NB be a good text classifier when its model of natural language is so oversimplified?
  - Even though the *probability estimates* of NB are of low quality, the *classification decisions* are surprisingly good

# Vector Space Classification

# 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

# Text Classification

- Naive Bayes classifier
  - Simple, cheap, high bias, linear
- K Nearest Neighbor classification
  - Simple, expensive at test time, high variance, non-linear
- Support Vector Machines (SVMs)
  - Soft margin SVMs and kernels for non-linear classifiers
- Any other classification model
- Some empirical evaluation and comparison
- Text-specific issues in classification

# Text Classification: Features

# 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
  - we use **all** of the words in the text (not a subset)

## 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

# 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 |
|---|---|---|---|---|---|
| $d_1$ | 0 | 0 | 1 | 1 | 1 |
| $d_2$ | 1 | 2 | 0 | 0 | 1 |
| $d_{test}$ | 0 | 1 | 0 | 0 | 1 |

| | Class |
|---|---|
| $d_1$ | + |
| $d_2$ | - |
| $d_{test}$ | ? |

This is the document-term matrix

# Doc-Term 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 |
|---|---|---|---|---|---|
| $d_1$ | 0 | 0 | 1 | 1 | 1 |
| $d_2$ | 1 | 1 | 0 | 0 | 1 |
| $d_{test}$ | 0 | 1 | 0 | 0 | 1 |

| | Class |
|---|---|
| $d_1$ | + |
| $d_2$ | - |
| $d_{test}$ | ? |

# 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 easily
- 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

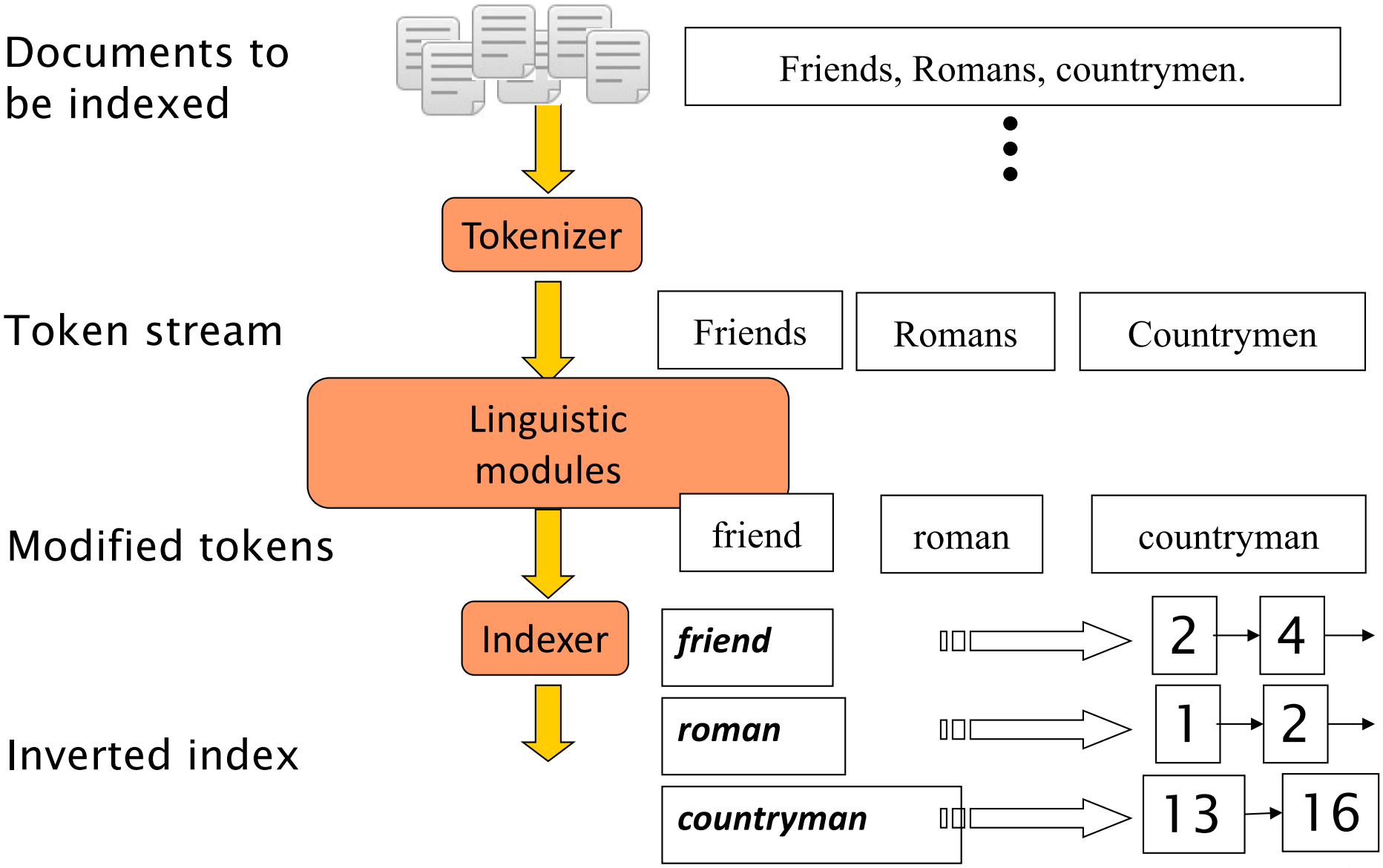
# Feature Selection: Frequency

- 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
- Smarter feature selection:
  - chi-squared, etc.

# Text pre-processing

Introduction to Information Retrieval

# Inverted index construction



# 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, …)

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 what granularity to use:

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: Tokenization

# 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

- 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. 20, 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,
      one answer is using n-grams: IIR ch. 3
  - Will often index “meta-data” separately
    - Creation date, format, etc.

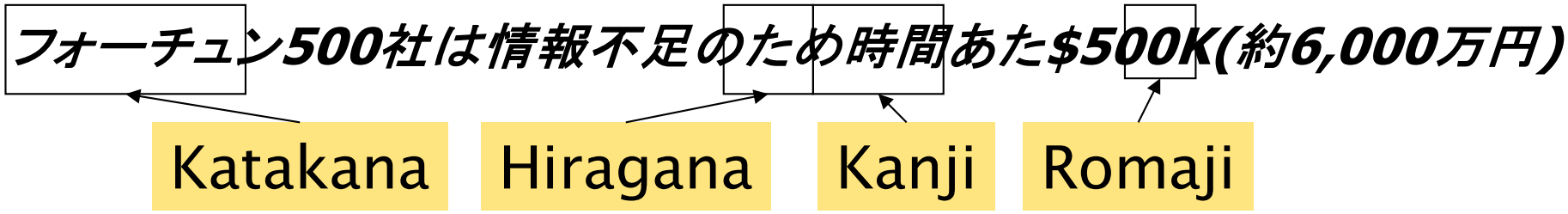
# Tokenization: language issues

- 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

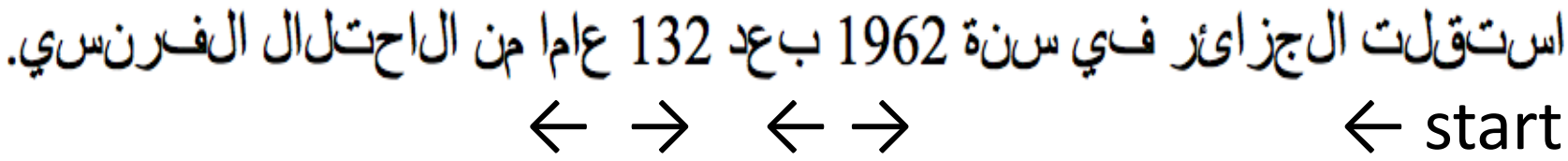
- Chinese and Japanese have no spaces between words:
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
  - Not always guaranteed a unique tokenization
- Further complicated in Japanese, with multiple alphabets intermingled
  - Dates/amounts in multiple formats




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




- ‘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

# Text Pre-processing: Stop Words & Normalization

# 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 (IIR 5) means the space for including stop words in a system is very small
  - Good query optimization techniques (IIR 7) 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***
  - deleting hyphens to form a term
    - ***anti-discriminatory, 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 likely 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***

# 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
  - 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.

# 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

# Text Pre-processing: Stemming

# Lemmatization

- Reduce inflectional/variant forms to base form
- E.g.,
  - *am, are, is* $\rightarrow$ *be*
  - *car, cars, car's, cars'* $\rightarrow$ *car*
- *the boy's cars are different colors* $\rightarrow$ *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 are both accept as equival to compress

# Porter's algorithm

- A 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.*

# Other stemmers

- Other stemmers exist:
  - Lovins stemmer
    - Single-pass, longest suffix removal (about 250 rules)
  - Paice/Husk stemmer
  - Snowball
- Full morphological analysis (lemmatization)
  - At most modest benefits for retrieval
- The above methods embody transformations that are
  - Language-specific, and often
  - Application-specific
- 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

- Definitely useful for Spanish, German, Finnish, …
  - 30% performance gains for Finnish!

# Text Classification: Real World Issues

# 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 IAAI 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 logistic regression/Neural networks/SVMs/random forests!

- 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 less practical

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

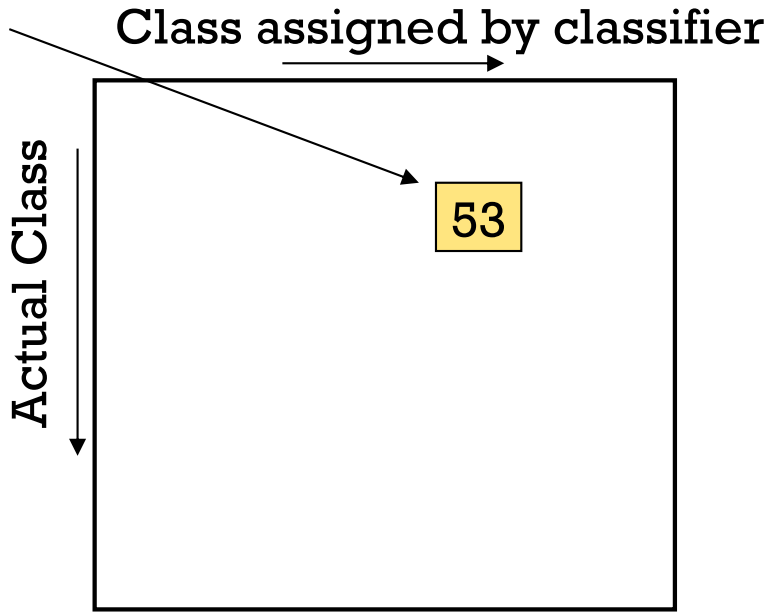
# How many categories?

- 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

# Good Practice: Inspect Confusion Matrix

- In a perfect classification, only the diagonal has non-zero entries
- Look at common confusions and how they might be addressed

  This (*i*, *j*) entry means 53 of the docs actually in class *i* were put in class *j* by the classifier.



# Good Practice: N-fold C.V.

- Results can vary based on sampling error due to different training and test sets.
- Average results over multiple training and test sets (splits of the overall data) for the best results.
- Ideally, test and training sets are independent on each trial.
  - But this would require too much labeled data.
- Partition data into N equal-sized disjoint segments.
- Run N trials, each time using a different segment of the data for testing, and training on the remaining N–1 segments.
- This way, at least test-sets are independent.
- Report average classification accuracy over the N trials.
- Typically, N = 10.

# Good Practice: Learning Curves

- In practice, labeled data is usually rare and expensive.
- Would like to know how performance varies with the number of training instances.
- *Learning curves* plot classification accuracy on independent test data (*Y* axis) versus number of training examples (*X* axis).
- One can do both the above and produce learning curves averaged over multiple trials from cross-validation

# 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! Easiest way to improve practical systems
    - Feature design and non-linear weighting is *very* important in the performance of real-world systems

# Upweighting

- You can get a lot of value by differentially weighting contributions from different document zones:
- That is, you count as two instances of a word when you see the word in, say, the abstract
  - Upweighting title words helps (Cohen & Singer 1996)
    - Doubling the weighting on the title words is a good rule of thumb
  - Upweighting the first sentence of each paragraph helps (Murata, 1999)
  - Upweighting sentences that contain title words helps (Ko *et al,* 2002)

# Does stemming/lowercasing/… help?

- As always, it's hard to tell, and empirical evaluation is normally the gold standard
- But note that the role of tools like stemming is rather different for TextCat vs. IR:
  - For IR, you often want to collapse forms of the verb *oxygenate* and *oxygenation*, since all of those documents will be relevant to a query for *oxygenation*
  - For TextCat, with sufficient training data, stemming *does no good.* It only helps in compensating for data sparseness (which can be severe in TextCat applications). *Overly aggressive stemming can easily degrade performance*

# 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."