Inspire…Educate…Transform.

NLP

Text mining and language understanding

Dr. K. V Dakshinamurthy

President, INSOFE

Text mining

- Class 1: Intro, thinking about the math behind text, text pre-processing, App-1: search
- Class 2: NB and classification (App-2), graphs (guest lectures)
- Class 3: EM and BBN
- Class 4: (App 3) Language modeling and Markov approximations, (App 4) Entity extraction tagging, HMM, Viterbi algorithm
- Class 5: (App 5) Sentiment extraction, App 6 (Spell Check)

Why is text so hard?

- It is non-linear
  - At last, a computer that understands you like your mother

1. (*) It understands you as well as your mother understands you
2. It understands (that) you like your mother
3. It understands you as well as it understands your mother

1 and 3: Does this mean well, or poorly?

Word sense ambiguity

At the semantic (meaning) level:

- They put money in the *bank*

  = buried in mud?

- I saw her duck with a telescope

Pronoun ambiguity

At the discourse (multi-clause) level:

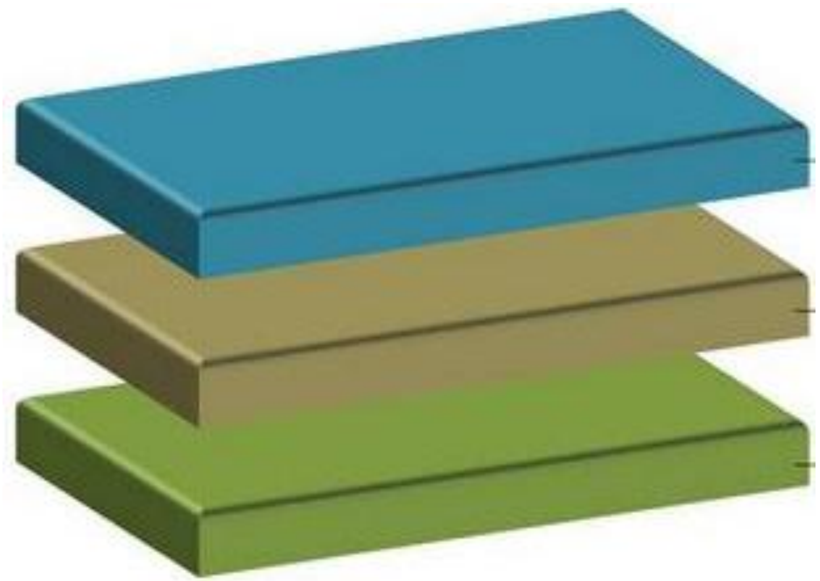
- Alice says they've built a computer that understands you like your mother
- But she . . .

  . . . doesn't know any details

  . . . doesn't understand me at all

This is an instance of anaphora, where she co-referees to some other discourse entity

Need to understand at different levels



- –Characters
- –Words
- –Phrases
- –Meanings
- –Hidden meanings
- –Styles

BASIC

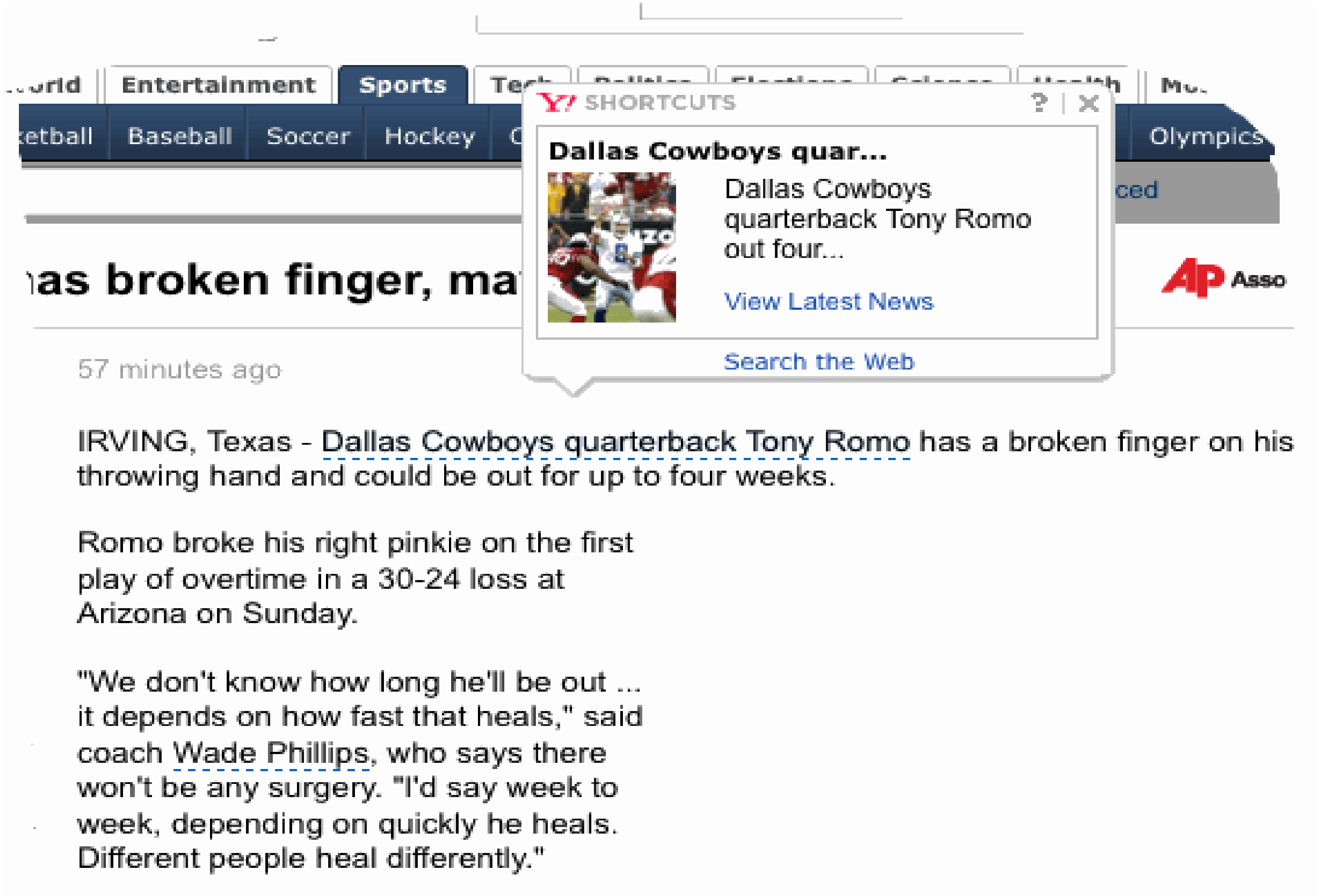
A few applications

- Detecting a language
  - Do I need to understand the
    - Meaning?
    - Words?
    - Letters?

Parts of speech tagging

- Chris made a mistake
  - Characters?
  - Words?
  - Phrases?

People, Places, Things



..orld | Entertainment | Sports | Tech | Politics | Elections | Science | Health | Mo..

..etball | Baseball | Soccer | Hockey | ... | Olympics

Y! SHORTCUTS

Dallas Cowboys quar...

Dallas Cowboys quarterback Tony Romo out four...

View Latest News

Search the Web

..as broken finger, ma..

AP Asso..

57 minutes ago

IRVING, Texas - Dallas Cowboys quarterback Tony Romo has a broken finger on his throwing hand and could be out for up to four weeks.

Romo broke his right pinkie on the first play of overtime in a 30-24 loss at Arizona on Sunday.

"We don't know how long he'll be out ... it depends on how fast that heals," said coach Wade Phillips, who says there won't be any surgery. "I'd say week to week, depending on quickly he heals. Different people heal differently."

APPLIED

Is this spam mail?

From: "" <takworlld@hotmail.com>
Subject: real estate is the only way... gem oalvgkay

Anyone can buy real estate with no money down

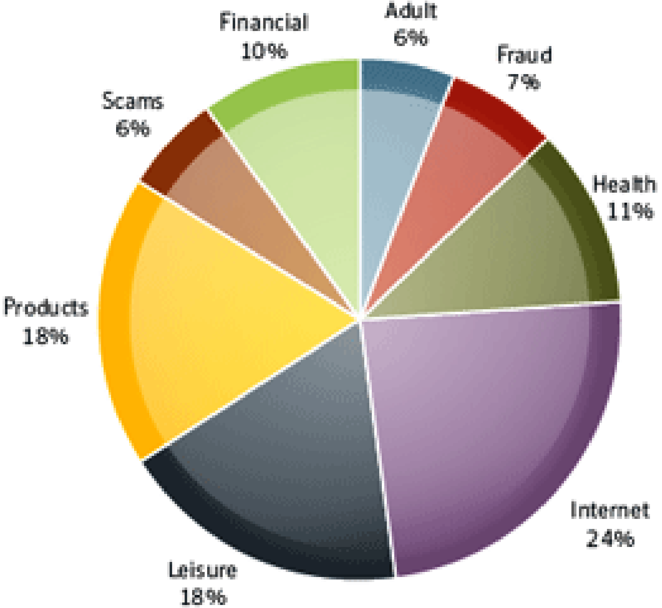
Stop paying rent TODAY !

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
=================================



Breakdown of Global Spam Categories – January 2009
Source: Symantec, Spam Monthly Report

News aggregation

Top Stories Personalized News Go

Stocks surge
CNNMoney.com - 1 hour ago
Dow jumps over 600 points before pulling back, reclaiming the 9000 level, as investors bet the worst is over. By Alexandra Twin, CNNMoney.
Stock markets rally on international push Bizjournals.com
6 Classic Signs of the Market's Bottom Motley Fool
The Canadian Press - Chicago Tribune - ABC News - Live 5 News
all 6,006 news articles »


The Associated Press

Treasury Department to name bailout manager Tuesday
Bizjournals.com - 53 minutes ago
The Treasury Department plans to name its prime contractor for the federal bailout job Tuesday, officials said Monday. "Congress passed the new law just 10 days ago, but in that time, we have accomplished a great deal on many fronts," said Neel ...
Federal Stakes in US Banks: Details, Please BusinessWeek
What Treasury is planning CNNMoney.com
Bloomberg - Forbes - guardian.co.uk - The Associated Press
all 1,202 news articles »

AFP

Search: Google's annual search statistics

Year	Annual Number of Google Searches	Average Searches Per Day
2013	2,161,530,000,000	5,922,000,000
2012	1,873,910,000,000	5,134,000,000
2011	1,722,071,000,000	4,717,000,000
2010	1,324,670,000,000	3,627,000,000
2009	953,700,000,000	2,610,000,000
2008	637,200,000,000	1,745,000,000
2007	438,000,000,000	1,200,000,000
2000	22,000,000,000	60,000,000
1998	3,600,000 **Googles official first year*	9,800

http://www.statisticbrain.com/google-searches/

Sentiment and Sarcasm

- My in-laws are as sweet as Nazis

– Great

  - “Delicious muesli from the @imaginarycafe- what a **great** way to start the day!
  - “**Greatly** disappointed that my local Imaginary Cafe have stopped stocking BLTs.”
  - “Had to wait in line for 45 minutes at the Imaginary Cafe today. **Great**, well there’s my lunchbreak gone…”

Examples of Language Processing Applications

Text Mining

Spam filtering

Document Classification

Date/time event detection

Information Extraction

(Web) Search engines

Information Retrieval

Watson in Jeopardy! (IBM)

Question Answering

Twitter brand monitoring

Sentiment Analysis (Stat. NLP)

Language Processing

Siri (Apple) and Google Now

Language Understanding

Spelling Correction

Statistical Language Modeling

Website translation (Google)

Machine Translation

"Clippy" Assistant (Microsoft)

Dialog System

Finding similar items (Amazon)

Recommender System

Learnings

- Not every text analytics requires understanding text
- Counting words can be powerful and sufficient at times

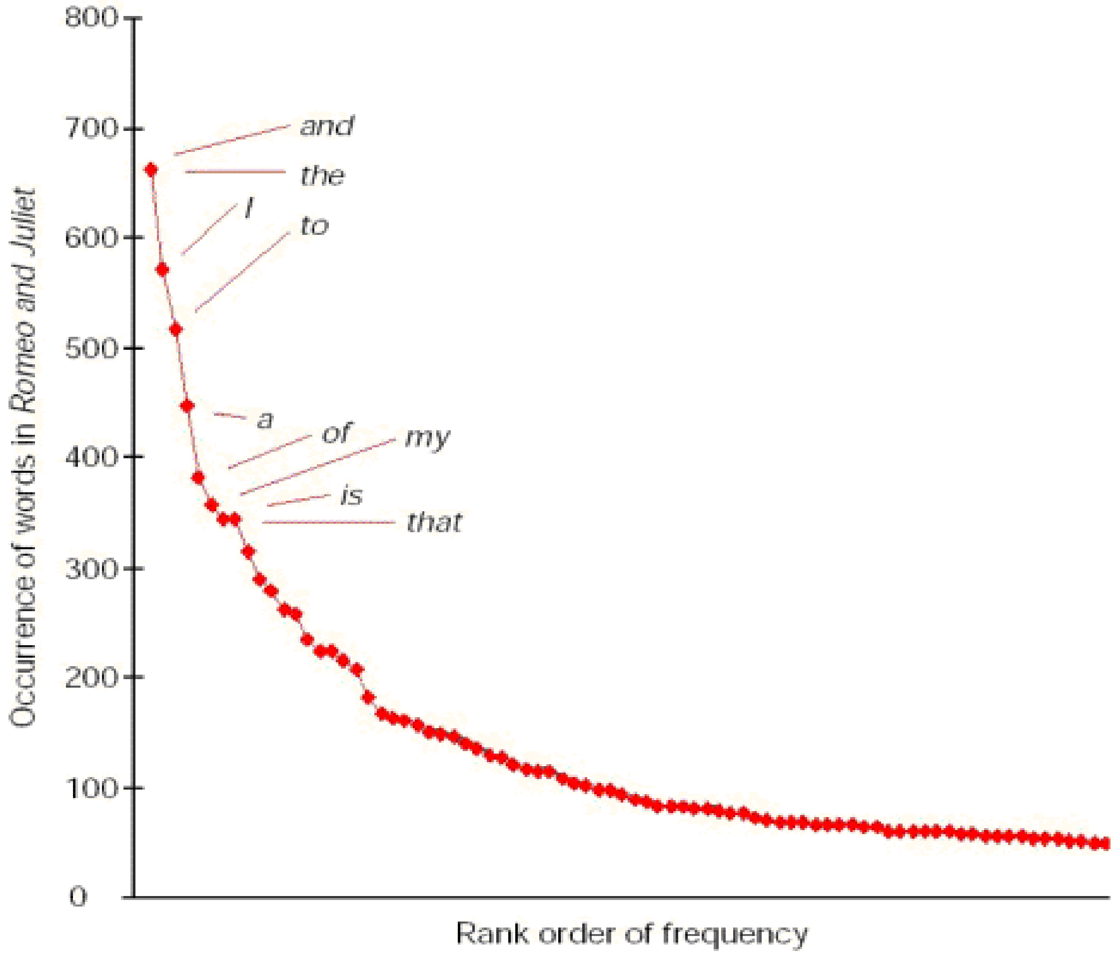
TEXT MINING: BASICS

Philosophy

- Statistical analyses of words
- Convert the words into a structured form
- Use statistics or data mining algorithms

LET US THINK ABOUT WORDS DIFFERENTLY

Words are not all equal



Sample Word Frequency Data

(from B. Croft, UMass)

Frequent Word	Number of Occurrences	Percentage of Total
the	7,398,934	5.9
of	3,893,790	3.1
to	3,364,653	2.7
and	3,320,687	2.6
in	2,311,785	1.8
is	1,559,147	1.2
for	1,313,561	1.0
The	1,144,860	0.9
that	1,066,503	0.8
said	1,027,713	0.8

Frequencies from 336,310 documents in the 1GB TREC Volume 3 Corpus
125,720,891 total word occurrences; 508,209 unique words

Text Property 1: Word Frequency

- A few words are very common.
  – 2 most frequent words (e.g. "the", "of") can account for about 10% of word occurrences.
- Most words are very rare.
  – Half the words in a corpus appear only once, called *hapax legomena* (Greek for "read only once")
- Called a "heavy tailed" distribution, since most of the probability mass is in the "tail"

Explanations for Zipf's Law

- Zipf's explanation was his "principle of least effort." Balance between speaker's desire for a small vocabulary and hearer's desire for a large one.

- Li (1992) shows that just random typing of letters including a space will generate "words" with a Zipfian distribution.
  - http://linkage.rockefeller.edu/wli/zipf/

Zipf's Law Impact on TM

- Good News: Stopwords will account for a large fraction of text so eliminating them greatly reduces storage costs.

- Bad News: For most words, gathering sufficient data for meaningful statistical analysis (e.g. for correlation analysis for query expansion) is difficult since they are extremely rare.

Words are not equally spaced

Spell correction

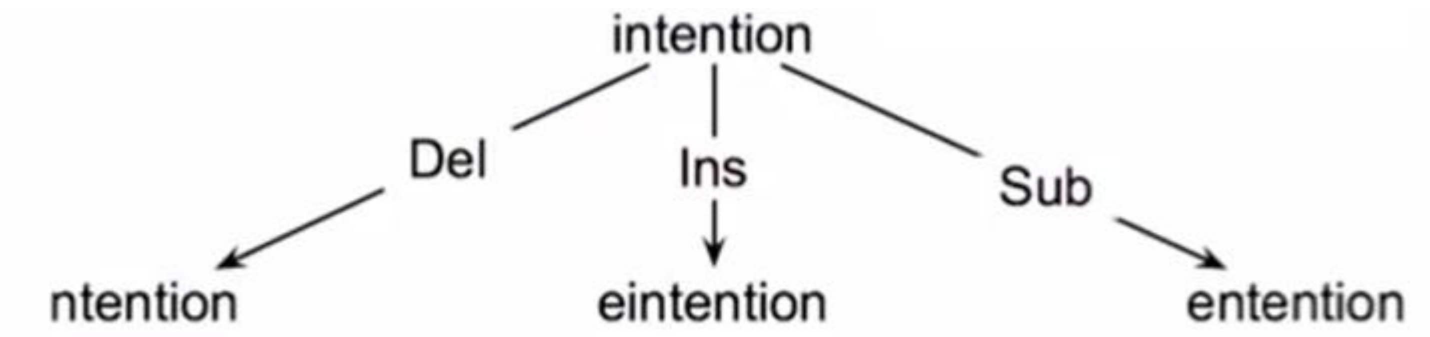
- The user typed “graffe”

Which is closest?

- graf
- graft
- grail
- giraffe

Minimum edit distance

- The minimum edit distance between two strings
- Is the minimum number of editing operations
  - Insertion
  - Deletion
  - Substitution
- Needed to transform one into the other


intention
Del
Ins
Sub
ntention
eintention
entention

MED

Two strings and their **alignment**:

```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N

D S S   I S
```

Distance is 5

MED

- Substitution is more involved and hence some researchers give higher cost to them than insertion and deletion. It is called Levinstein distance.

What is the MED and LD of

- CATS to BAT

- C -> B 2 in LD and 1 in MED
- A -> A
- T -> T
- S -> _ 1

- So, the distance is 3 in LD or 2 in MED

Computational biology also use MED

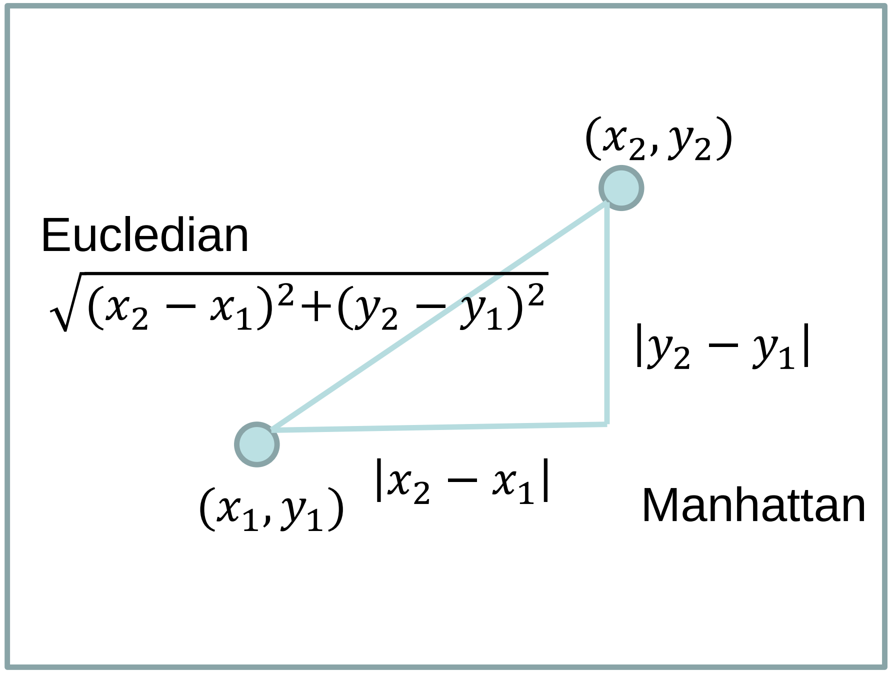
Given a sequence of bases

AGGCTATCACCTGACCTCCAGGCCGATGCCC
TAGCTATCACGACCGCGGTCGATTTGCCCGAC

Searching for a path (sequence of edits) from the start string to the final string:

- **Initial state**: the word we're transforming
- **Operators**: insert, delete, substitute
- **Goal state**: the word we're trying to get to
- **Path cost**: what we want to minimize: the number of edits

Elements of Coordinate geometry



Distance for numeric attributes

- Minkowski distance: h is positive integer.

$$dist(\mathbf{x}_i, \mathbf{x}_j) = ((x_{i1} - x_{j1})^h + (x_{i2} - x_{j2})^h + \ldots + (x_{ir} - x_{jr})^h)^{\frac{1}{h}}$$

h= 2 is Euclidian and h=1 is Manhattan distance

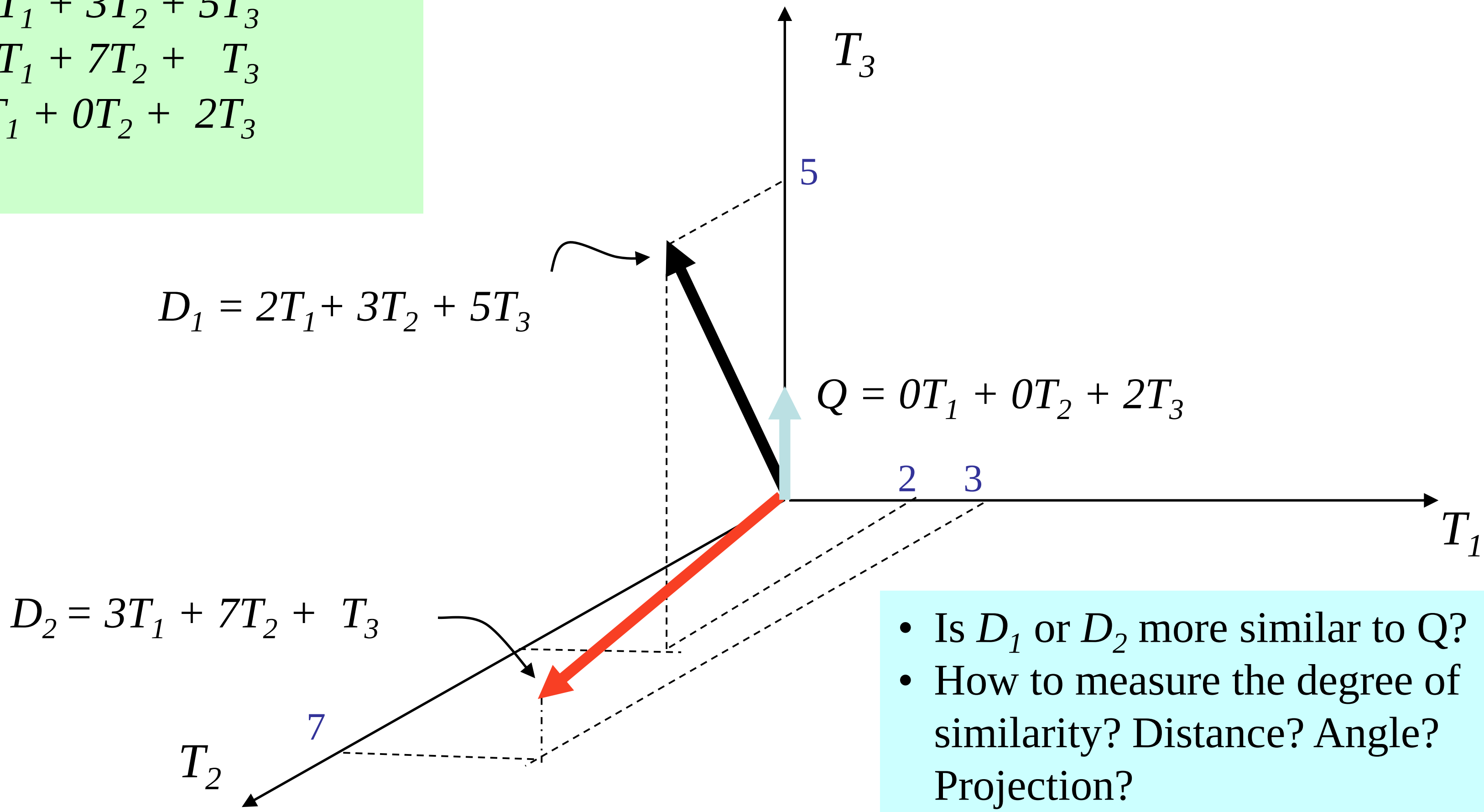
Vector Space Representation

Example:

$D_1 = 2T_1 + 3T_2 + 5T_3$

$D_2 = 3T_1 + 7T_2 + T_3$

$Q = 0T_1 + 0T_2 + 2T_3$



- Is D_1 or D_2 more similar to Q?
- How to measure the degree of similarity? Distance? Angle? Projection?

Similarity Measure

- We now have vectors for all documents in the collection, a vector for the query, how to compute similarity?

- Using a similarity measure between the query and each document:

  - It is possible to rank the retrieved documents in the order of presumed relevance (query-dependent ranking).

  - It is possible to enforce a certain threshold so that the size of the retrieved set can be controlled.

Desiderata for proximity

- If d_1 is near d_2, then d_2 is near d_1.
- If d_1 near d_2, and d_2 near d_3, then d_1 is not far from d_3.
- No document is closer to d than d itself.

First cut: Euclidean distance

- Distance between vectors d_1 and d_2 is the length of the vector $|d_1 – d_2|$.
  – Euclidean distance

- Why is this not a great idea?

- Length normalization
  – Long documents will be more similar to each other by virtue of length, not topic

- Second Cut: Same problem with Manhattan distance.

- We could implicitly normalize by looking at *angles* instead

Third cut: Inner Product

- Similarity between vectors for the document $\boldsymbol{d}_i$ and query $\boldsymbol{q}$ can be computed as the vector inner product:

$$\text{sim}(\boldsymbol{d}_j,\boldsymbol{q}) = \boldsymbol{d}_\text{j}\bullet\boldsymbol{q} = \sum_{i=1}^{t} w_{ij} \cdot w_{iq}$$

where w_{ij} is the weight of term i in document j and w_{iq} is the weight of term i in the query

- For binary vectors, the inner product is the number of matched query terms in the document (size of intersection)

- For weighted term vectors, it is the sum of the products of the weights of the matched terms.

Properties of Inner Product

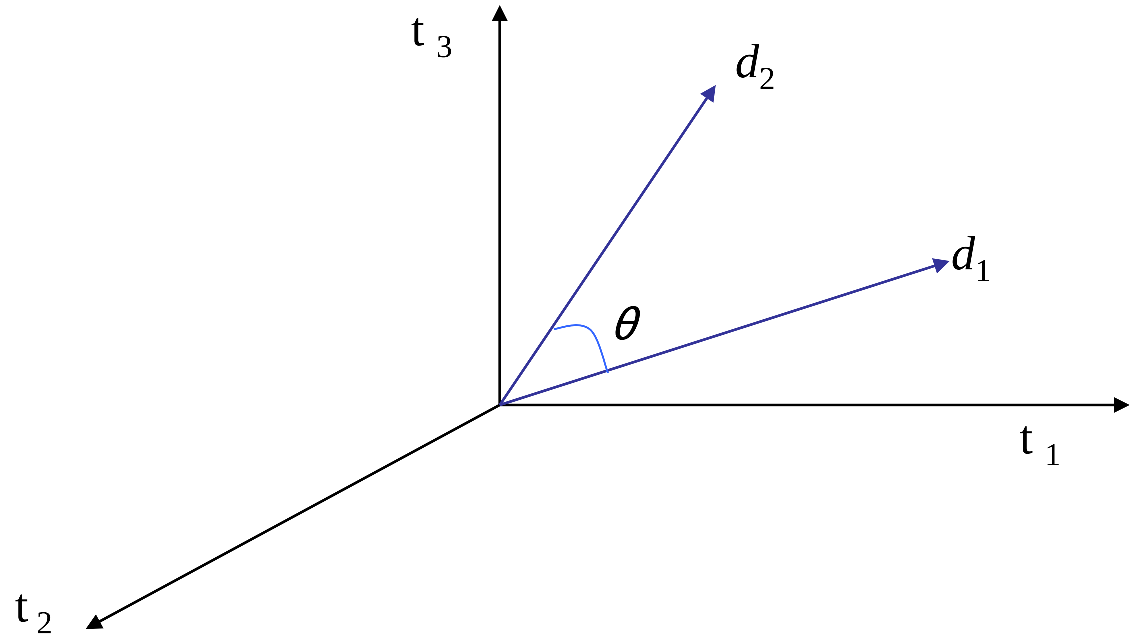
- Favors long documents with a large number of unique terms.
  - Again, the issue of length normalization

- Measures how many terms matched but not how many terms are *not* matched.

Cosine similarity

- Distance between vectors d_1 and d_2 *captured* by the cosine of the angle x between them.

- Note – this is actually *similarity*, not distance



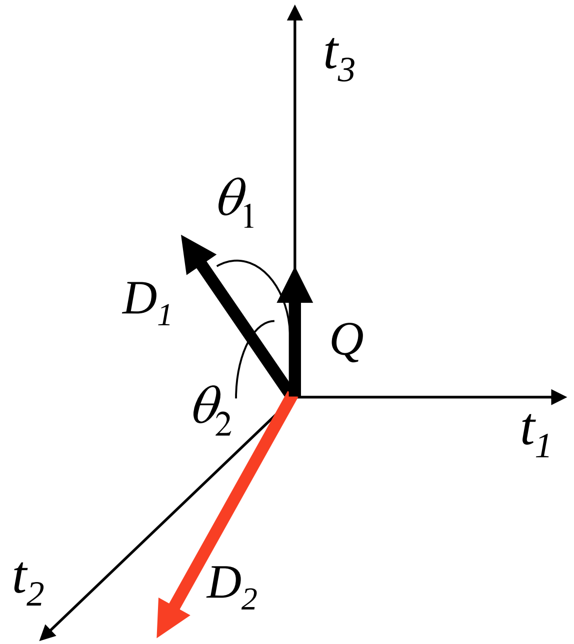
Cosine similarity

$$sim(d_j, d_k) = \frac{\vec{d}_j \cdot \vec{d}_k}{\left|\vec{d}_j\right|\left|\vec{d}_k\right|} = \frac{\sum_{i=1}^{n} w_{i,j} w_{i,k}}{\sqrt{\sum_{i=1}^{n} w_{i,j}^2}\sqrt{\sum_{i=1}^{n} w_{i,k}^2}}$$

- Cosine of angle between two vectors
- The denominator involves the lengths of the vectors
- The cosine measure is also known as the *normalized inner product*

Cosine Similarity vs. Inner Product

$D_1 = 2T_1 + 3T_2 + 5T_3$ CosSim(D_1 , Q) = 10 / √(4+9+25)(0+0+4) = 0.81
$D_2 = 3T_1 + 7T_2 + 1T_3$ CosSim(D_2 , Q) = 2 / √(9+49+1)(0+0+4) = 0.13
$Q = 0T_1 + 0T_2 + 2T_3$



D_1 is 6 times better than D_2 using cosine similarity but only 5 times better using inner product.

Cosine similarity exercise

- *Exercise: Rank the following by decreasing cosine similarity:*
  - Two documents that have only frequent words ***(the, a, an, of)*** in common.
  - Two documents that have no words in common.
  - Two documents that have many rare words in common ***(wingspan, tailfin).***

Words have fixed coordinates!!!

- Based on how words are occurring w.r.t other words, can we fix absolute positions of all words?
  - Google's word2vec
  - Every word is assigned with a 30 dimensional coordinate
  - https://code.google.com/p/word2vec/

Summary

- Words have different densities
- Different distances
- Actually, fixed coordinates! 

TEXT PREPROCESSING

Tokenization

- Analyze text into a sequence of discrete tokens (words).
- Sometimes punctuation (“e-mail”, “a.out”), numbers (“1999”), and case (“Congress” vs. “congress”) can be a meaningful part of a token.
  - However, frequently they are not.
- Simplest approach: ignore all numbers and punctuation and use only case-insensitive unbroken strings of alphabetic characters as tokens.
- More careful approach:
  - Separate ? ! ; : “ ‘ [ ] ( ) < >
  - Care with .
  - Care with other punctuation marks

Tokenization (continued)

- Example: “We’re attending a tutorial now.” → we ’re attending a tutorial now

- Downloadable tool:
  - Word Splitter
    http://l2r.cs.uiuc.edu/~cogcomp/atool.php?tkey=WS

Numbers

- 3/12/91
- Mar. 12, 1991
- 55 B.C.
- B-52
- 100.2.86.144
  - Generally, don't index as text

Case Folding

- Reduce all letters to lower case
  - exception: upper case in mid-sentence
    - *e.g.*, ***General Motors***
    - ***Fed*** vs. ***fed***
    - ***SAIL*** vs. ***sail***

Tokenizing HTML

- Should text in HTML commands not typically seen by the user be included as tokens?
  - Words appearing in URLs.
  - Words appearing in “meta text” of images.

- Simplest approach is to exclude all HTML tag information (between “<“ and “>”) from tokenization. But could lose critical information.

Stopwords

- Stop words are language & domain dependent
- For efficiency, store strings for stop words in a hash table to recognize them in constant time.

- How to determine a list of stopwords?
  - For English? – may use existing lists of stopwords
    - E.g. SMART's common word list
    - WordNet stopword list
    - http://www.ranks.nl/resources/stopwords.html
    - http://ir.dcs.gla.ac.uk/resources/linguistic_utils/stop_words
  - For Spanish? Bulgarian? Hindi?

Stemming

- Reduce tokens to "root" form of words to recognize morphological variation.
  - "computer", "computational", "computation" all reduced to same token "compute"
- Correct morphological analysis is language specific and can be complex.
- Stemming "blindly" strips off known affixes (prefixes and suffixes) in an iterative fashion.

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

→

for exampl compres and compres are both accept as equival to compres.

Porter Stemmer – Sample Rules

http://tartarus.org/~martin/PorterStemmer/

- remove ending
  - if a word ends with a consonant other than s, followed by an s, then delete s.
  - if a word ends in es, drop the s.
  - if a word ends in ing, delete the ing unless the remaining word consists only of one letter or of th.
  - If a word ends with ed, preceded by a consonant, delete the ed unless this leaves only a single letter.
  - ……
- transform words
  - if a word ends with "ies" but not "eies" or "aies" then "ies --> y."

Porter Stemmer

- Simple procedure for removing known affixes in English without using a dictionary.
- Can produce unusual stems that are not English words:
  - "computer", "computational", "computation" all reduced to same token "comput"
- May conflate (reduce to the same token) words that are actually distinct.
- Does not recognize all morphological derivations.

Other stemmers

- Other stemmers exist, e.g., Lovins stemmer
- Single-pass, longest suffix removal (about 250 rules)
- Full morphological analysis – only modest benefits for retrieval

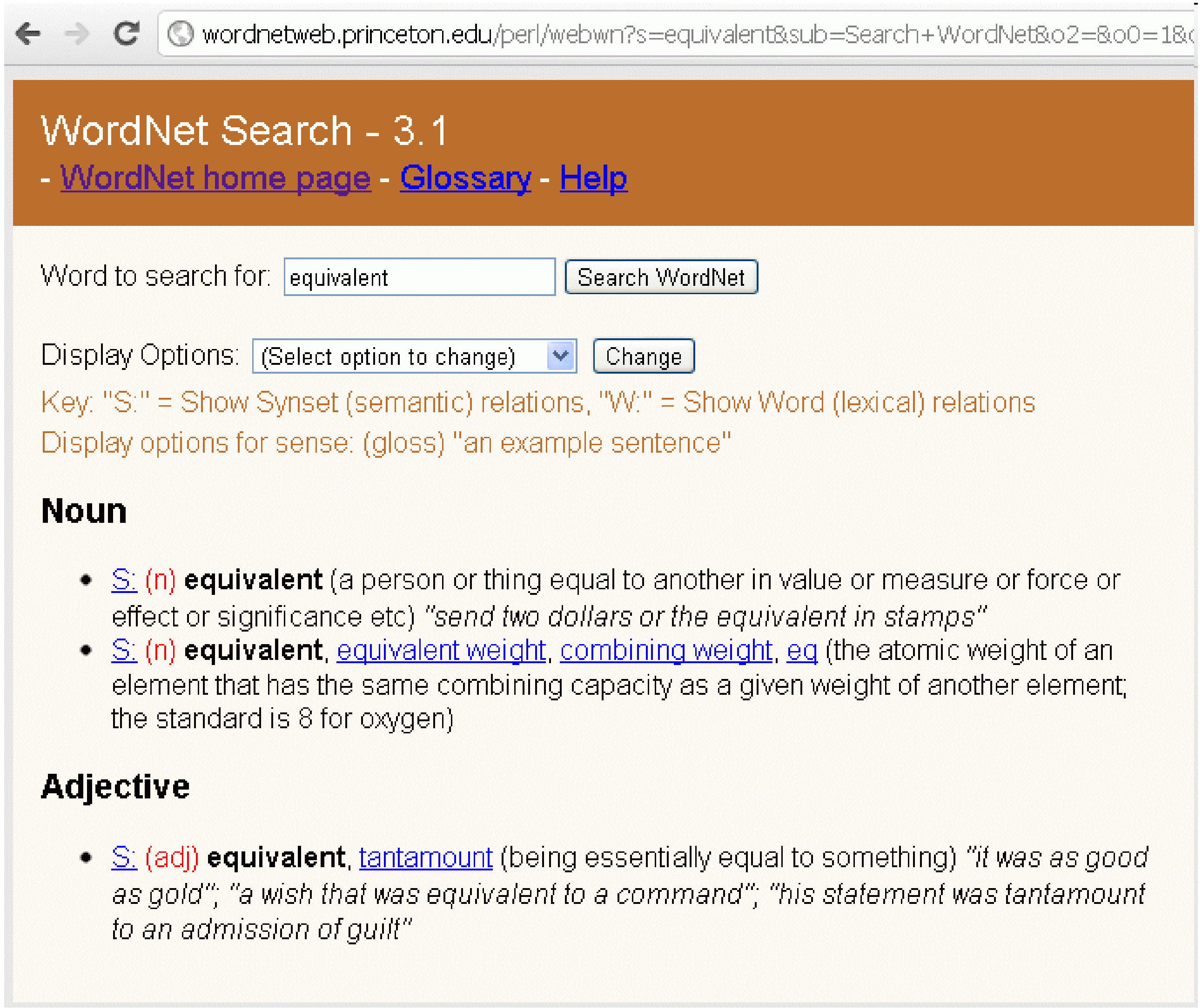
Lemmatization

- Lemmatization implies a broader scope of fuzzy word matching that is still handled by the same subsystems. It implies certain techniques for low level processing within the engine, and may also reflect an engineering preference for terminology.
- A lemmatization system would handle matching “car” to “cars” along with matching “car” to “automobile”. In a more traditional search engine, matching “car” to “cars” would be handled by stemming, but matching “car” to “automobile” would be handled by a separate system.
- Practical implementation: use WordNet’s morphstr function

WordNet

WordNet® is a large, <u>free</u> lexical database of English.

Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept.



Search

APPLICATION 1

Steps

- Pre-processing
- Inverted index
- Retreival
- Ranking

STEP 2: INVERTED INDEX

Term-document incidence

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

Goal: ***Brutus*** *AND* ***Caesar*** *BUT NOT* ***Calpurnia***

1 if play contains word,
0 otherwise

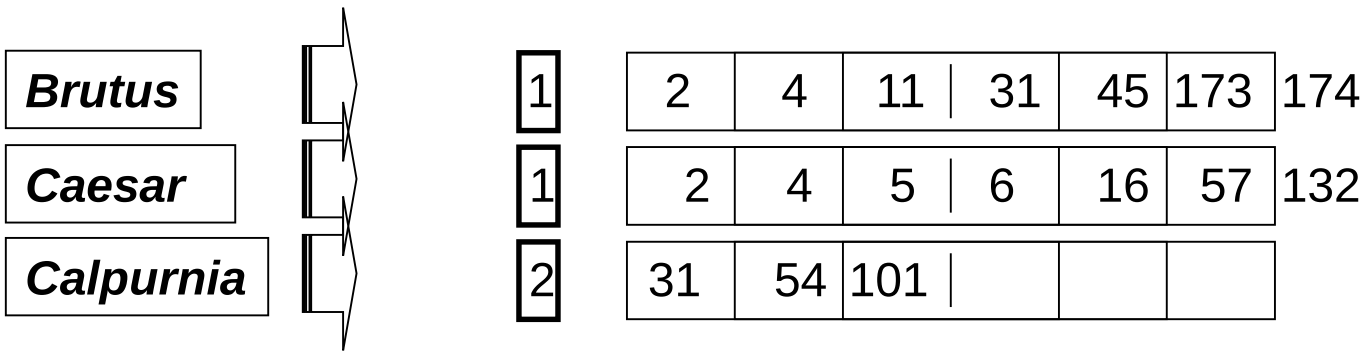
Inverted Index: Motivation

Sec. 1.1

- *N* = 1 million documents, each with about 1000 words.
- *M* = 500K *distinct* terms among these.
- 500K x 1M matrix has half-a-trillion 0's and 1's.
- But it has no more than one billion 1's
  - matrix is extremely sparse.
- A better representation?
  - Only record the 1 positions.

Inverted index

- For each term *t*, store a list of all documents that contain *t*.
  - Identify each by a **docID**, a document serial number
  - Optionally store the number & position of occurrence of t in d.

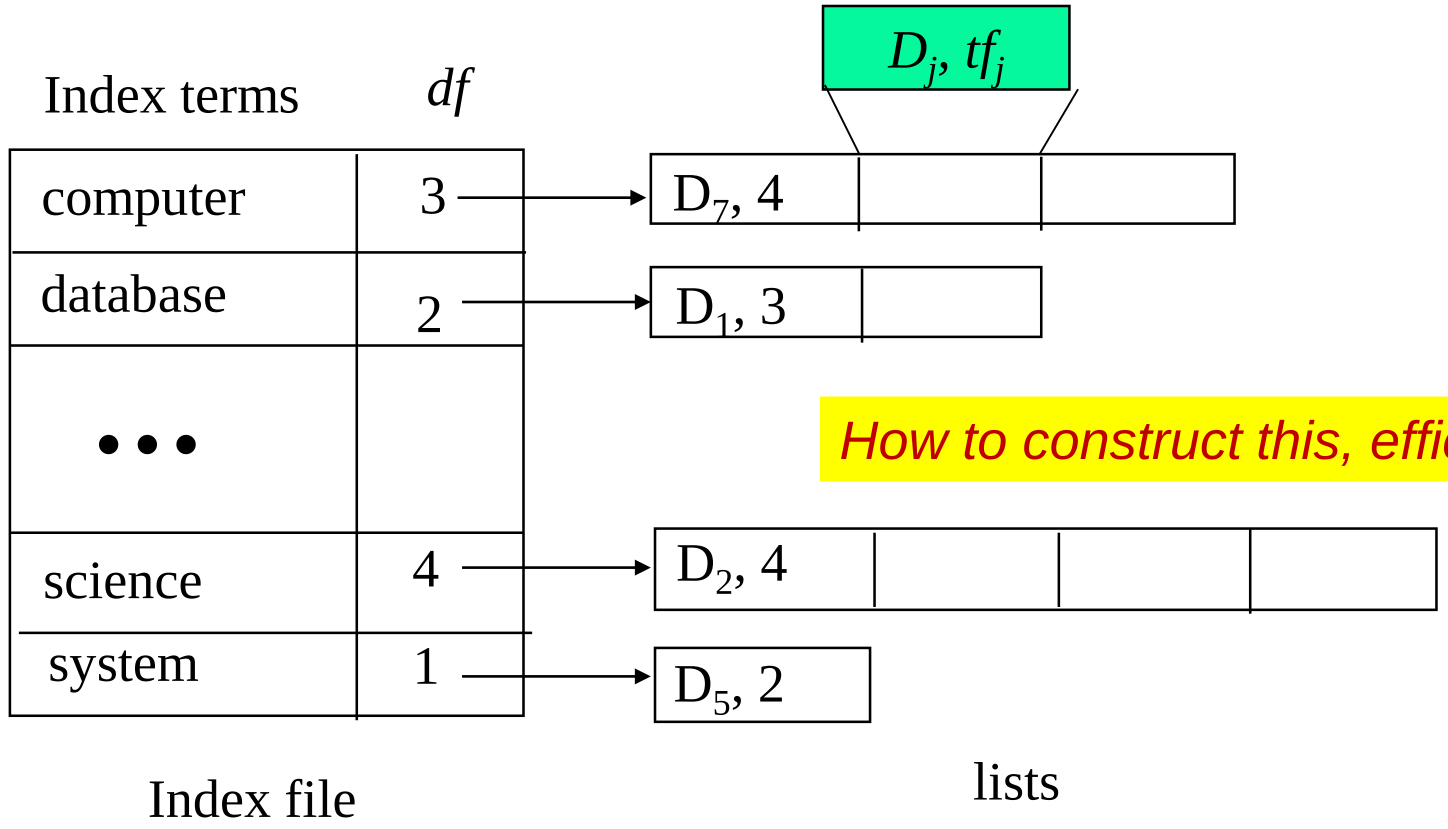


Cannot use fixed-size arrays for this.

Hence, post-facto insertion is an issue. This is best done as a batch job.

Inverted Index – in practice

- We need:
  - One entry for each word in the vocabulary
  - For each such entry:
    - Keep a list of all the documents where it appears together with the corresponding frequency → TF
  - For each such entry, keep the total number of occurrences in all documents:
    - IDF



How to construct this, efficiently?

Indexer steps: Token sequence

- Sequence of (Modified token, Document ID) pairs.

Doc 1	Doc 2
I did enact Julius Caesar I was killed i' the Capitol; Brutus killed me.	So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious

Term	docID
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

Indexer steps: Sort

- Sort by terms
  - And then docID

Core indexing step

Term	docID
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

Term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

Indexer steps: Dictionary & Postings

- Multiple term entries in a single document are merged.
- Split into Dictionary and Postings
- Document frequency information is added.

Term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

term	doc. freq.	→	postings lists
ambitious	1	→	2
be	1	→	2
brutus	2	→	1 → 2
capitol	1	→	1
caesar	2	→	1 → 2
did	1	→	1
enact	1	→	1
hath	1	→	2
i	1	→	1
i'	1	→	1
it	1	→	2
julius	1	→	1
killed	1	→	1
let	1	→	2
me	1	→	1
noble	1	→	2
so	1	→	2
the	2	→	1 → 2
told	1	→	2
you	1	→	2
was	2	→	1 → 2
with	1	→	2

Where is the storage requirement?

term	doc. freq.	→	postings lists
ambitious	1	→	2
be	1	→	2
brutus	2	→	1 → 2
capitol	1	→	1
caesar	2	→	1 → 2
did	1	→	1
enact	1	→	1
hath	1	→	2
i	1	→	1
i'	1	→	1
it	1	→	2
julius	1	→	1
killed	1	→	1
let	1	→	2
me	1	→	1
noble	1	→	2
so	1	→	2
the	2	→	1 → 2
told	1	→	2
you	1	→	2
was	2	→	1 → 2
with	1	→	2

Terms and counts

Lists of docIDs

Pointers

Inverted Index: Another Example

Doc 1	Doc 2	Doc 3	Doc 4
one fish, two fish	red fish, blue fish	cat in the hat	green eggs and ham

	1	2	3	4
blue		1		
cat			1	
egg				1
fish	1	1		
green				1
ham				1
hat			1	
one	1			
red		1		
two	1			

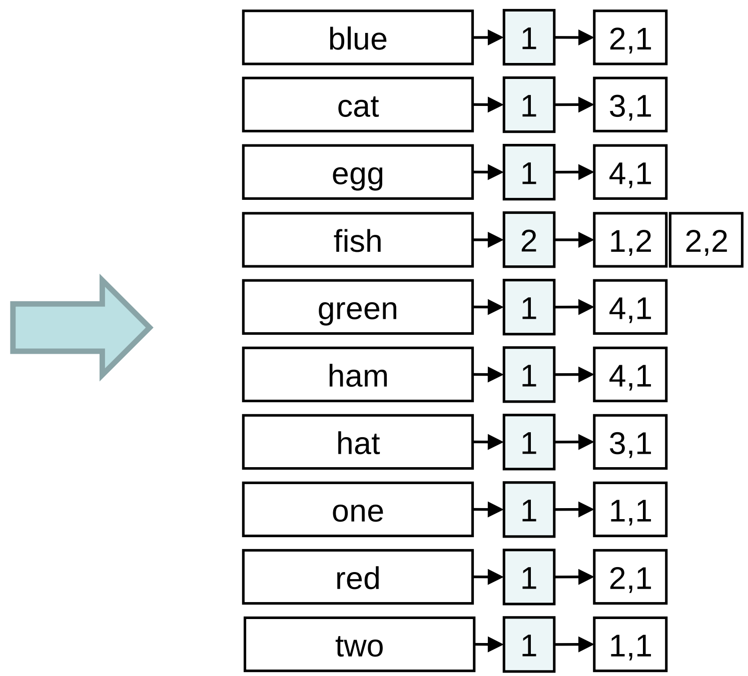
blue →	2	
cat →	3	
egg →	4	
fish →	1	2
green →	4	
ham →	4	
hat →	3	
one →	1	
red →	2	
two →	1	

Inverted Index: Ranked Retrieval

Doc 1	Doc 2	Doc 3	Doc 4
one fish, two fish	red fish, blue fish	cat in the hat	green eggs and ham

tf

	1	2	3	4	*df*
blue		1			1
cat			1		1
egg				1	1
fish	2	2			2
green				1	1
ham				1	1
hat			1		1
one	1				1
red		1			1
two	1				1



Inverted Index: Positional Information

Doc 1	Doc 2	Doc 3	Doc 4
one fish, two fish	red fish, blue fish	cat in the hat	green eggs and ham

blue	1	2,1	
cat	1	3,1	
egg	1	4,1	
fish	2	1,2	2,2
green	1	4,1	
ham	1	4,1	
hat	1	3,1	
one	1	1,1	
red	1	2,1	
two	1	1,1	

blue	1	2,1,[3]	
cat	1	3,1,[1]	
egg	1	4,1,[2]	
fish	2	1,2,[2,4]	2,2,[2,4]
green	1	4,1,[1]	
ham	1	4,1,[3]	
hat	1	3,1,[2]	
one	1	1,1,[1]	
red	1	2,1,[1]	
two	1	1,1,[3]	

Indexing: Performance Analysis

- Fundamentally, a large sorting problem
  - Terms usually fit in memory
  - Postings usually don't
- How is it done on a single machine?
- How can it be done with MapReduce?
- First, let's characterize the problem size:
  - Size of vocabulary
  - Size of postings

Query processing: AND

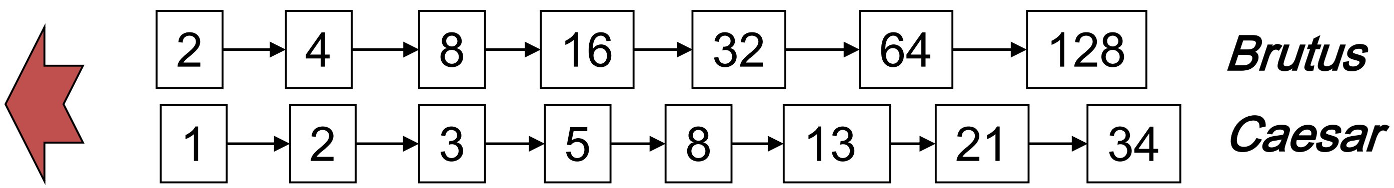
- Consider processing the query:

  Brutus *AND* ***Caesar***

  – Locate ***Brutus*** in the Dictionary;
    - Retrieve its postings.

  – Locate *Caesar* in the Dictionary;
    - Retrieve its postings.

  – "Merge" the two postings:



STEP 3 & 4: RETRIEVAL & RANKING

Comments on Vector Space Models

- Simple, mathematically based approach.
- Considers both local (*tf*) and global (*idf*) word occurrence frequencies.
- Provides partial matching and ranked results.
- Tends to work quite well in practice despite obvious weaknesses.
- Allows efficient implementation for large document collections.

Problems with Vector Space Model

- Missing syntactic information (e.g. phrase structure, word order, proximity information).
- Missing semantic information (e.g. word sense).
- Assumption of term independence (e.g. ignores synonomy).
- Lacks the control of a Boolean model (e.g., *requiring* a term to appear in a document).
  - Given a two-term query “A B”, may prefer a document containing A frequently but not B, over a document that contains both A and B, but both less frequently.

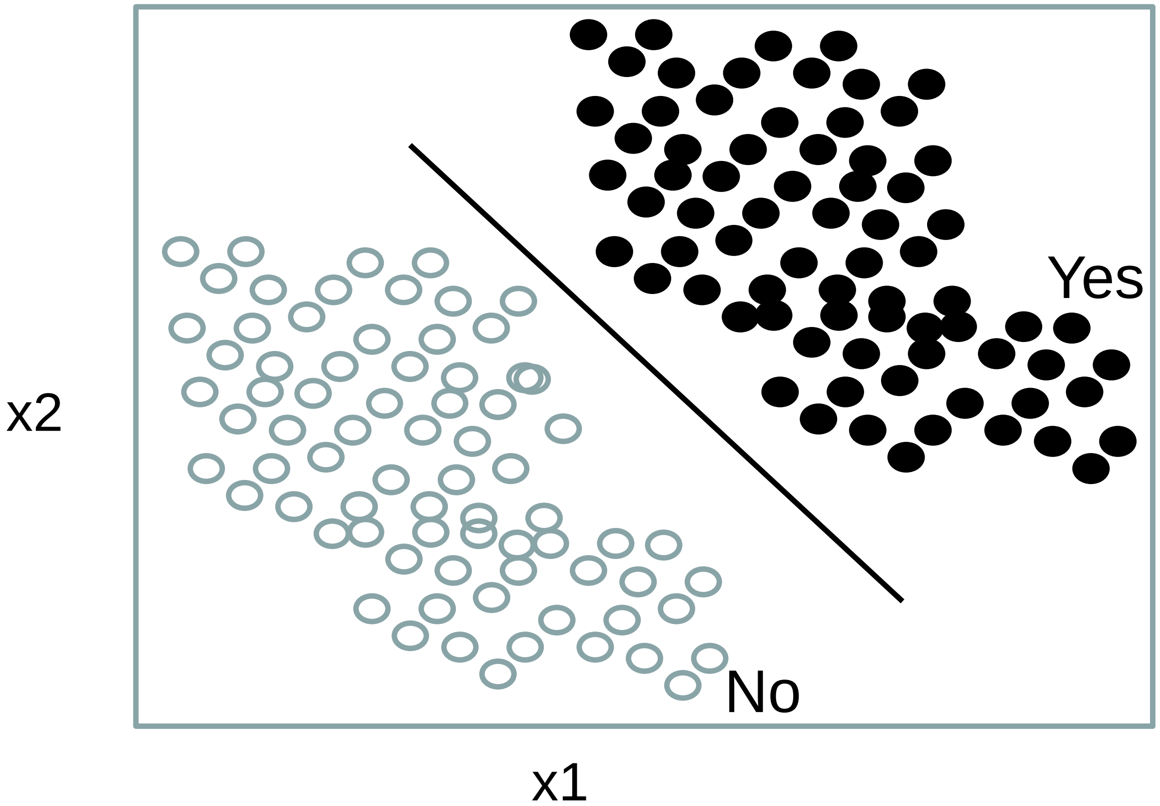
NAÏVE BAYES

Supervised data: Historic

x1	x2	x3	x4	x5	y
Easy to measure attributes					Difficult to measure

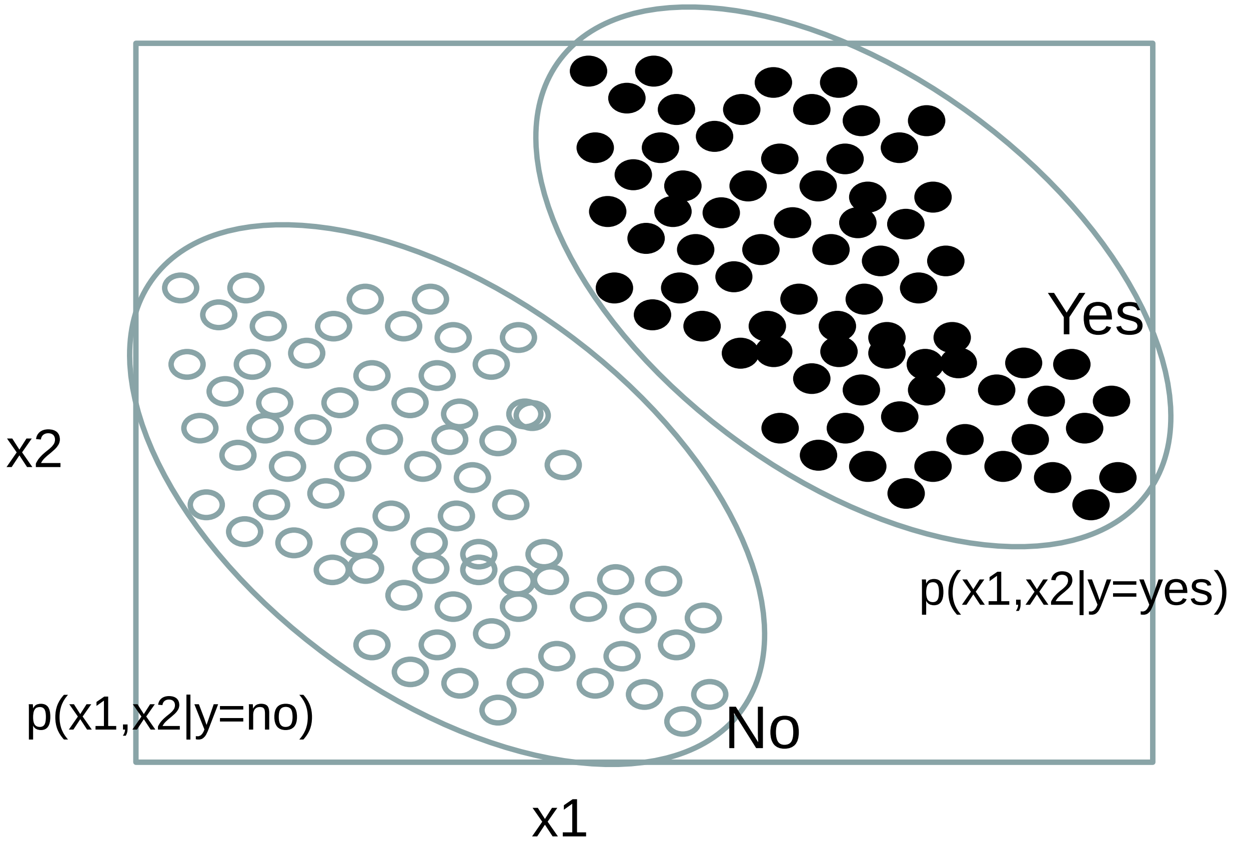
Can we derive a function f such that y=f(x1,x2,x3,x4,x5) using historic data
So that, we can predict y for every new value of x

Mutiple ways to do this



Logistic regression

Multiple ways to do this



Generative model

- Compute join distribution
- Can generate new samples
- Generative models are naturally adaptive
- Generative models can listen to data and take your existing knowledge

What you measure

- $p(x1, x2, \ldots | y{=}no)$

- But, what you need to predict in future is
  - $p(y{=}no | x1, x2\ldots)$

Bayes theorem

$$p(y = no|x1, x2) = \frac{p(x1, x2|y = no).p(y = no)}{p(x1, x2)}$$

P(y=no) is prior belief

Left hand side is how belief is updated with evidence

Denominator is ignored as it is same for all classes

Compute LHS for all classes and pick the class with highest p

Bayesian Classification

- Problem statement:
  - Given features $X_1, X_2, \ldots, X_n$
  - Predict a label Y from a set of known labels
  - i.e., Compute $\arg\max_{Y} P(Y | X_1, \ldots, X_n)$

The Bayes Classifiers

- Use Bayes Rule!

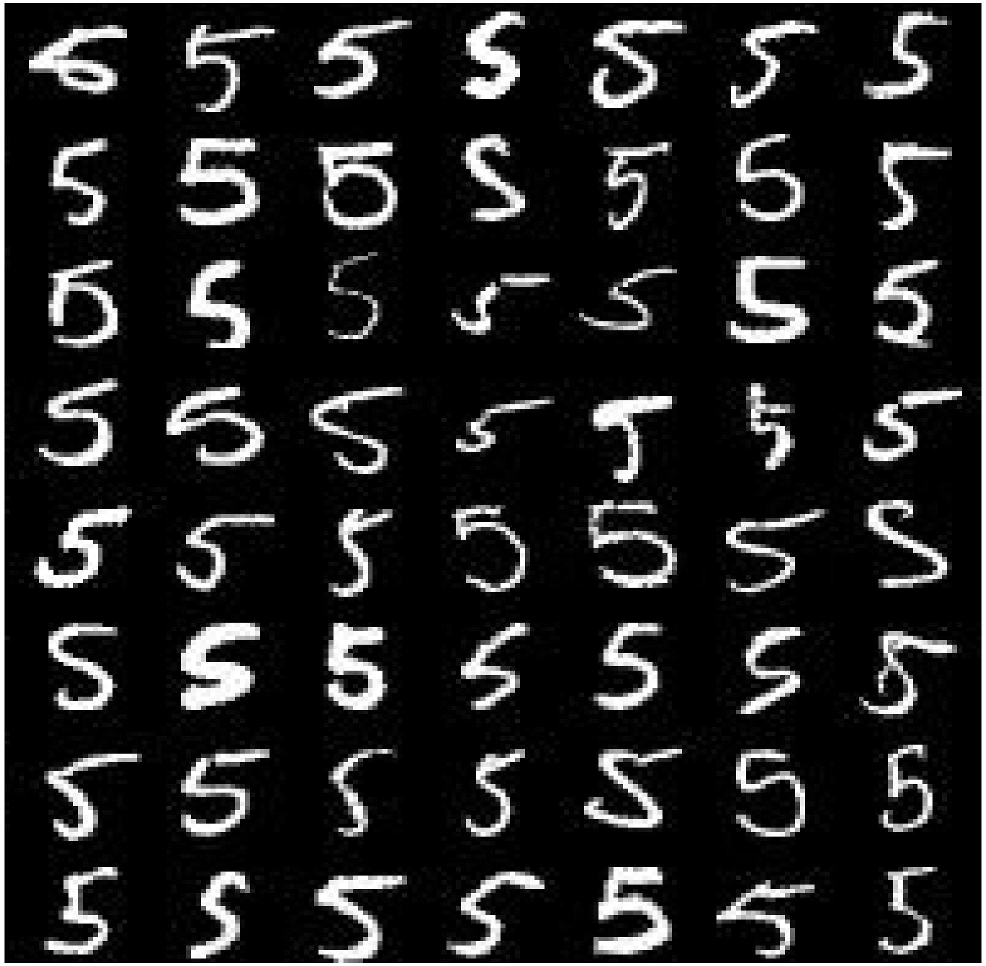
Likelihood

Prior

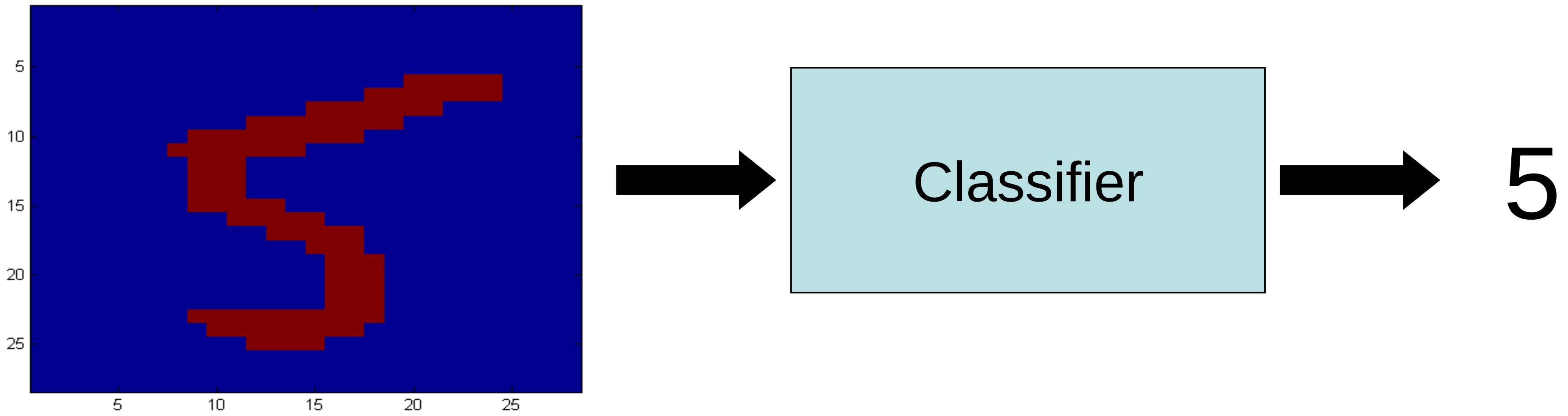
$$P(Y|X_1,\ldots,X_n) = \frac{P(X_1,\ldots,X_n|Y)P(Y)}{P(X_1,\ldots,X_n)}$$

Normalization Constant

An Imaging Application

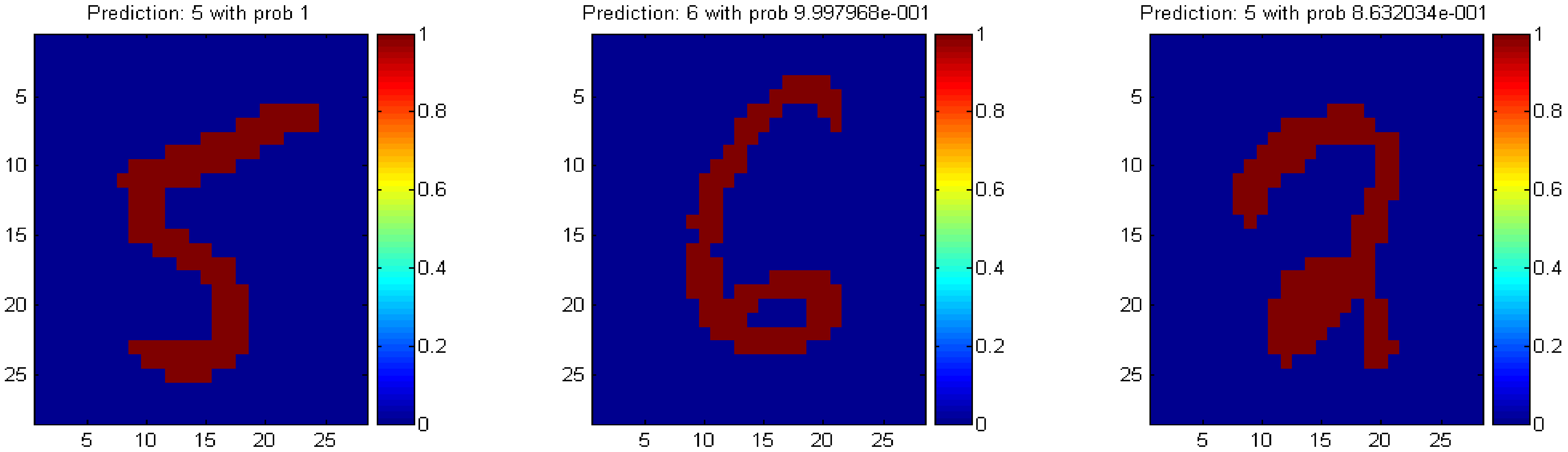


Digit Recognition (contd.)



- $X_1,\ldots,X_n \in \{0,1\}$ **(Black vs. White pixels)**
- $Y \in \{5,6\}$ **(predict whether a digit is a 5 or a 6)**

Test Cases


Prediction: 5 with prob 1
Prediction: 6 with prob 9.997968e-001
Prediction: 5 with prob 8.632034e-001

The Bayes Classifier

- Let's expand this for our digit recognition task:

$$P(Y=5|X_1,\ldots,X_n) = \frac{P(X_1,\ldots,X_n|Y=5)\,P(Y=5)}{P(X_1,\ldots,X_n|Y=5)P(Y=5)+P(X_1,\ldots,X_n|Y=6)P(Y=6)}$$

$$P(Y=6|X_1,\ldots,X_n) = \frac{P(X_1,\ldots,X_n|Y=6)\,P(Y=6)}{P(X_1,\ldots,X_n|Y=5)P(Y=5)+P(X_1,\ldots,X_n|Y=6)P(Y=6)}$$

- To classify, we'll simply compute these two probabilities and predict whichever is greater.

The Naïve Bayes Assumption

- Assume that all features are independent **given the class label Y**
- Mathematically:

$$P(X_1, \ldots, X_n|Y) = \prod_{i=1}^{n} P(X_i|Y)$$

- Validity of this assumption?

Why is this useful?

- # of parameters for modeling $P(X_1,\ldots,X_n|Y)$:
  - $2(2^n-1)$
- # of parameters for modeling $P(X_1|Y),\ldots,P(X_n|Y)$
  - $2n$

Another Illustration

Training sample pairs (X, D)

X = (x_1, x_2, …, x_n) is the feature vector representing the instance.

n = 4

x_1= outlook = {sunny, overcast, rain}

x_2= temperature = {hot, mild, cool}

x_3= humidity = {high, normal}

x_4= wind = {weak, strong}

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	*No*
Day2	Sunny	Hot	High	Strong	*No*
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	*No*
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	*No*
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	*No*

An Illustration

Training sample pairs (X, D)

D=Play Tennis = {yes, no}

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	*No*
Day2	Sunny	Hot	High	Strong	*No*
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	*No*
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	*No*
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	*No*

Bayesian Classifier

- The Bayesian approach to classifying a new instance X is to assign it to the most probable target value Y (MAP classifier)

$$
\begin{aligned}
Y &= \arg\max_{d_i \in d} p(d_i \mid X) \\
&= \arg\max_{d_i \in d} p(d_i \mid x_1, x_2, x_3, x_4) \\
&= \arg\max_{d_i \in d} \frac{p(x_1, x_2, x_3, x_4 \mid d_i) P(d_i)}{p(x_1, x_2, x_3, x_4)} \\
&= \arg\max_{d_i \in d} p(x_1, x_2, x_3, x_4 \mid d_i) P(d_i)
\end{aligned}
$$

Bayesian Classifier

$$Y = \arg\max_{d_i \in d} p(x_1, x_2, x_3, x_4 \mid d_i) P(d_i)$$

$P(d_i)$ is easy to calculate: simply counting how many times each target value d_i occurs in the training set

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	*No*
Day2	Sunny	Hot	High	Strong	*No*
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	*No*
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	*No*
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	*No*

P(d = yes) = 9/14

P(d = no) = 5/14

Bayesian Classifier

$$Y = \arg\max_{d_i \in d} p(x_1, x_2, x_3, x_4 \mid d_i) P(d_i)$$

- $P(x_1, x_2, x_3, x_4|d_i)$ is much more difficult to estimate.
- In this simple example, there are 3x3x2x2x2 = 72 possible terms
- To obtain a reliable estimate, we need to see each terms many times
- Hence, we need a very, very large training set! (which in most cases is impossible to get)

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

Naïve Bayes Assumption

Attribute values are conditionally independent given the target value.

This means, we have

$$\boxed{P(x_1, x_2, \cdots, x_n \mid d_i) = \prod_i P\left(x_i \mid d_i\right)}$$

Naïve Bayes Classifier

$$\boxed{Y \quad = \arg\max_{d_i \in d} P(d_i) \prod_{k=1}^{4} P(x_k \mid d_i)}$$

Back to the Example

Naïve Bayes Classifier

$$Y = \arg\max_{d_i \in d} \prod_{k=1}^{4} P(x_k \mid d_i) P(d_i)$$

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	*No*
Day2	Sunny	Hot	High	Strong	*No*
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	*No*
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	*No*
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	*No*

$$Y = \arg\max_{d_i \in \{yes, no\}} P(d_i) P(x_1 = suny \mid d_i) P(x_2 = cool \mid d_i) P(x_3 = high \mid d_i) P(x_4 = strong \mid d_i)$$

Play-tennis example: estimating $P(x_i|C)$

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

outlook	
P(sunny|p) = 2/9	P(sunny|n) = 3/5
P(overcast|p) = 4/9	P(overcast|n) = 0
P(rain|p) = 3/9	P(rain|n) = 2/5
temperature	
P(hot|p) = 2/9	P(hot|n) = 2/5
P(mild|p) = 4/9	P(mild|n) = 2/5
P(cool|p) = 3/9	P(cool|n) = 1/5
humidity	
P(high|p) = 3/9	P(high|n) = 3/5
P(normal|p) = 6/9	P(normal|n) = 2/5
windy	
P(true|p) = 3/9	P(true|n) = 3/5
P(false|p) = 6/9	P(false|n) = 2/5

P(p) = 9/14
P(n) = 5/14

Back to the Example

$$Y = \arg\max_{d_i \in \{yes,no\}} P(d_i)P(x_1 = suny \mid d_i)P(x_2 = cool \mid d_i)P(x_3 = high \mid d_i)P(x_4 = strong \mid d_i)$$

P(d=yes)=9/14 = 0.64

P(d=no)=5/14 = 0.36

$P(x_1$=sunny|yes)=2/9
$P(x_1$=sunny|no)=3/5

$P(x_2$=cool|yes)=3/9
$P(x_2$=cool|no)=1/5

$P(x_3$=high|yes)=3/9
$P(x_3$=high|no)=3/5

$P(x_4$=strong|yes)=3/9
$P(x_4$=strong|no)=3/5

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

Back to the Example

$$Y \quad = \arg\max_{d_i \in \{yes,no\}} P(d_i)P(x_1 = suny \mid d_i)P(x_2 = cool \mid d_i)P(x_3 = high \mid d_i)P(x_4 = strong \mid d_i)$$

$$P(yes)P(x_1 = suny \mid yes)P(x_2 = cool \mid yes)P(x_3 = high \mid yes)P(x_4 = strong \mid yes) = 0.0053$$

$$P(no)P(x_1 = suny \mid no)P(x_2 = cool \mid no)P(x_3 = high \mid no)P(x_4 = strong \mid no) = 0.0206$$

Y = Play Tennis = no

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	*No*
Day2	Sunny	Hot	High	Strong	*No*
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	*No*
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	*No*
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	*No*

A Note on Estimating Probabilities

- So far, we estimate the probabilities by the fraction of times the event is observed to occur over the entire opportunities

- In the above example, we estimated

  P(wind=strong|play tennis=no)=Nc/N,

  where N = 5 is the total number of training samples for which play tennis = no,
  Nc is the number of these for which wind=strong

- What happens if Nc =0 or too small?

Note.. continued

- When Nc is small, however, such approach provides poor estimation. To avoid this difficulty, we can adopt the **m-estimate** of probability

$$\boxed{\frac{N_C + mP}{N + m}}$$

where P is the prior estimate of the probability we wish to estimate, m is a constant called the equivalent sample size.

A typical method for choosing P in the absence of other information is to assume uniform priors: If an attribute has k possible values we set P=1/K.

For example, P(wind =strong | play tennis=no), we note wind has two possible values, so uniform priors means P = ½

Naïve Bayes in Email Spam filtering

SPAM FILTERING

- Suppose we wanted to build a spam filter. To use the "bag of words" approach, assuming that *n* words in an email are **conditionally independent,** we'd get:

$$P(spam|\mathbf{w}) \propto \prod_{i=1}^{n} \hat{P}(w_i|spam)\hat{P}(spam)$$

$$P(\neg spam|\mathbf{w}) \propto \prod_{i=1}^{n} \hat{P}(w_i|\neg spam)\hat{P}(\neg spam)$$

- Whichever one's bigger wins!

- *Training data:* a corpus of email messages, each message annotated as spam or no spam.
  - *Task:* classify new email messages as spam/no spam.

Training: estimate priors

$$P(v_j) = \frac{n}{N}$$

Estimate likelihoods using the **m-estimate:**

$$P(w_k|v_j) = \frac{n_k+1}{n+|Vocabulary|}$$

N: total number of words in all emails

n: number of words in emails with class v_j

n_k: number of times word w_k occurs in emails with class v_j

$|Vocabulary|$: size of the vocabulary

Testing: to classify a new email, assign it the class with the highest posterior probability. Ignore unknown words.

Classify_naive_Bayes_text(*Doc*)

1. *positions*: all word positions in *Doc* that contain tokens found in *Vocabulary*
2. Return ω_{NB} , where

$$\omega_{NB} = \arg\max_{\omega_j} P(\omega_j) \prod_{i \text{ in } positions} P(x_i \mid \omega_j)$$

Summary of Naïve Bayes

- Bayes' rule can be turned into a classifier
- Maximum A Posteriori (MAP) hypothesis estimation incorporates prior knowledge; Max Likelihood doesn't
- Naive Bayes Classifier is a simple but effective Bayesian classifier for vector data (i.e. data with several attributes) that assumes that attributes are independent given the class.
- Bayesian classification is a generative approach to classification

International School of Engineering

Plot 63/A, 1st Floor, Road # 13, Film Nagar, Jubilee Hills, Hyderabad - 500 033

For Individuals: +91-9502334561/63 or 040-65743991

For Corporates: +91-9618483483

Web: http://www.insofe.edu.in

Facebook: https://www.facebook.com/insofe

Twitter: https://twitter.com/Insofeedu

YouTube: http://www.youtube.com/InsofeVideos

SlideShare: http://www.slideshare.net/INSOFE

LinkedIn: http://www.linkedin.com/company/international-school-of-engineering

This presentation may contain references to findings of various reports available in the public domain. INSOFE makes no representation as to their accuracy or that the organization subscribes to those findings.