

# Agenda

---

- Introduction to Text Mining
- Binary term incidence matrix
- Information Retrieval Pipeline
- Inverted Index Construction
- Merge Algorithm and Query Optimization
- Tolerant Retrieval using Normalization, Query expansion, Stemming, Wild card query using K-Gram index
- Ranked Retrieval using TF-IDF
- Python Implementation

# A Brief History of Web Search

---

- 1991: Tim Berners-Lee “invents” the World Wide Web
  - First Web search engines:
    - Archie: Query file names by regular expressions
    - Architext/Excite: Full text search, simple ranking (1993)
  - Until 1998, web search meant information retrieval
  - 1998: Google was founded – Exploits link structure using the PageRank algorithm
-

# Jargons

➤ Corpus /Collections



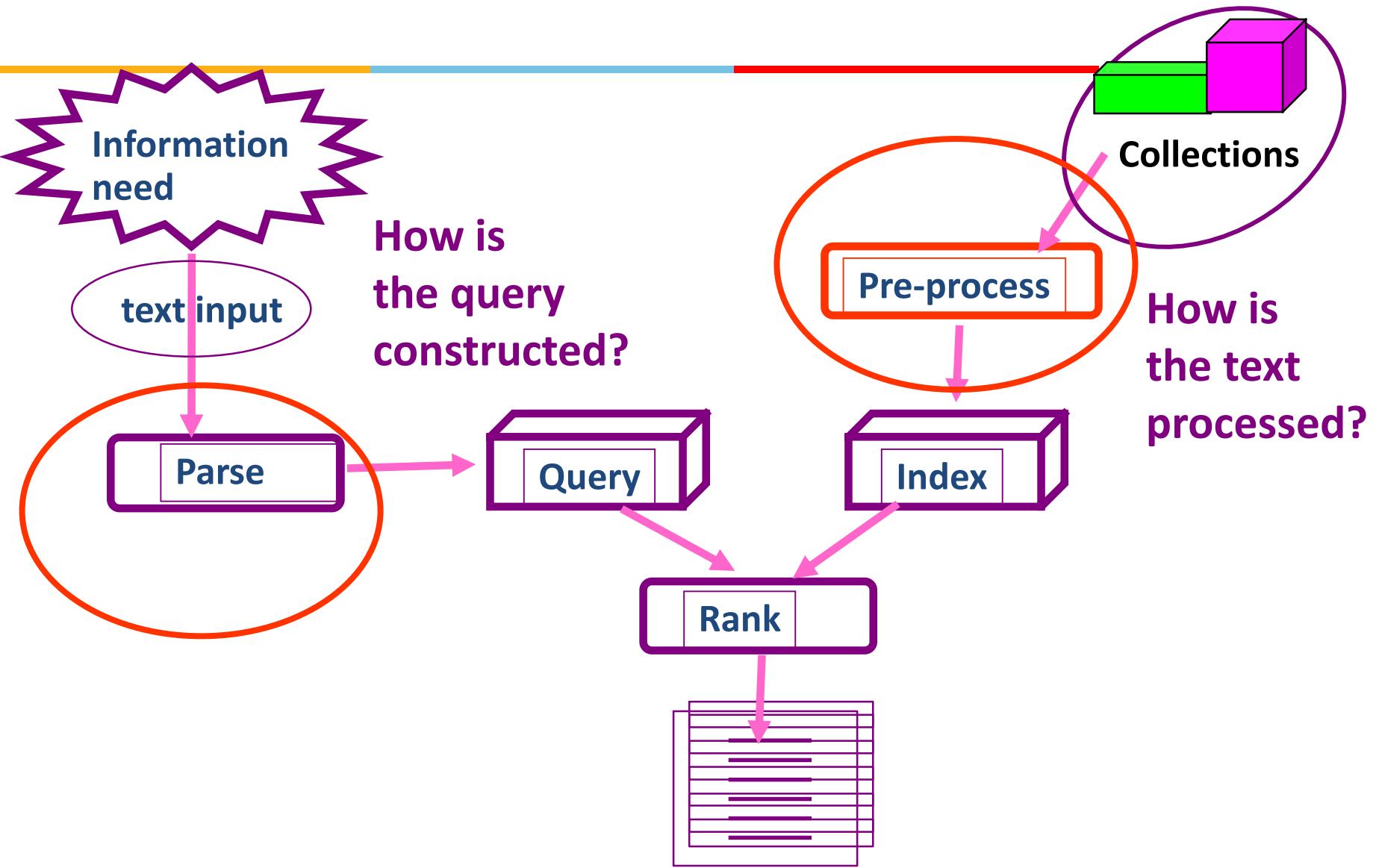
➤ Information Need



➤ Query



# Information Retrieval Process



# Bag of Words representation

---

- A very popular and basic representation of documents is the bag of words model.
- Each document is represented by a bag (= multiset) of terms from a predefined vocabulary.

# Bag of Words representation



# The Jackal was eyeing at the grapes

# He was as cunning as a Jackal

These Grapes are too sweet but the poor Jackal could not have it.

<b>1</b>	<b>2</b>	<b>0</b>		<b>1</b>			<b>1</b>			<b>1</b>			<b>1</b>					
a	as	at		Cunning			he			was			jackal					

# Term Incidence Matrix

	T1	T2	T3	T4	T5	T6
D1	1	0	0	1	1	0
D2	0	1	0	1	1	0
D3	1	0	1	0	1	1
D4	1	0	1	0	1	1

	T1	T2	T3	T4	T5	T6
D1	6	0	0	2	1	0
D2	0	8	0	5	3	0
D3	2	0	6	0	5	2
D4	5	0	2	0	6	7

# Boolean queries: Exact match

---

- The Boolean Retrieval model is being able to ask a query that is a Boolean expression:
  - Boolean Queries are queries using *AND*, *OR* and *NOT* to join query terms
    - Views each document as a set of words
    - Is precise: document matches condition or not.
- Some professional searchers (e.g., lawyers) still like Boolean queries.

# Example: WestLaw

<http://www.westlaw.com/>

---

Largest commercial (paying subscribers) legal search service (started 1975; ranking added 1992)

Tens of terabytes of data; 700,000 users

Majority of users *still* use boolean queries

# Inverted index construction

Documents to be indexed.

Token stream.

**Tokenizer**

Linguistic modules

Modified tokens.

**Indexer**

Inverted index.

Friends

Romans

Countrymen

friend

friend

roman

countryman

roman

countryman

2 → 4

1 → 2

13 → 16

# Indexer steps

Sequence of (Modified token, Document ID) pairs.

**Doc 1**

I did enact Julius  
Caesar I was killed  
i' the Capitol;  
Brutus killed me.

**Doc 2**

So let it be with  
Caesar. The noble  
Brutus hath told you  
Caesar was ambitious



Term	Doc #
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

- Frequency information is added.

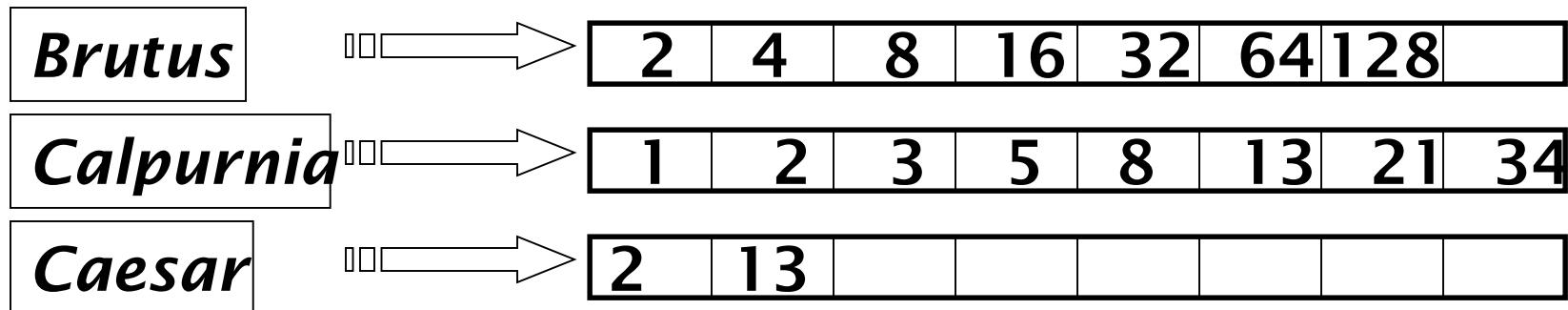
Term	Doc #
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2



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

# Inverted index

For each term  $T$ , we must store a list of all documents that contain  $T$ .



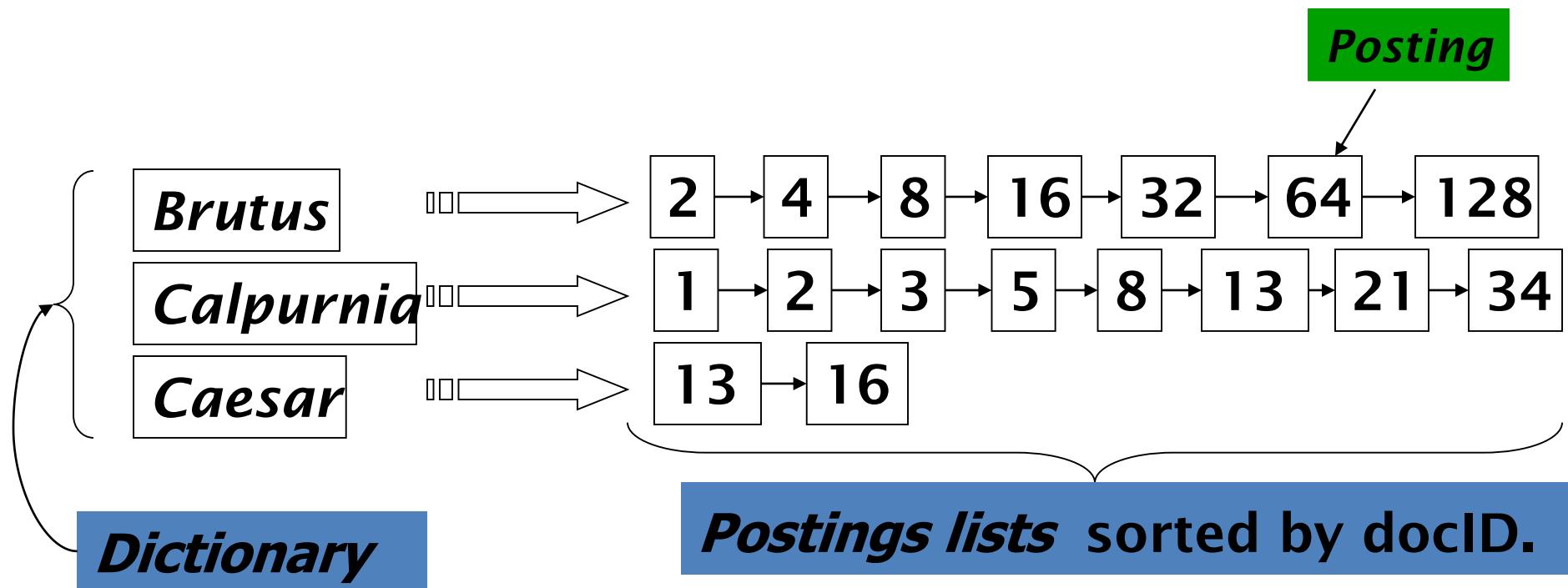
- we can use an array or a linked list.

What happens if the word ***Caesar*** is added to document 14?

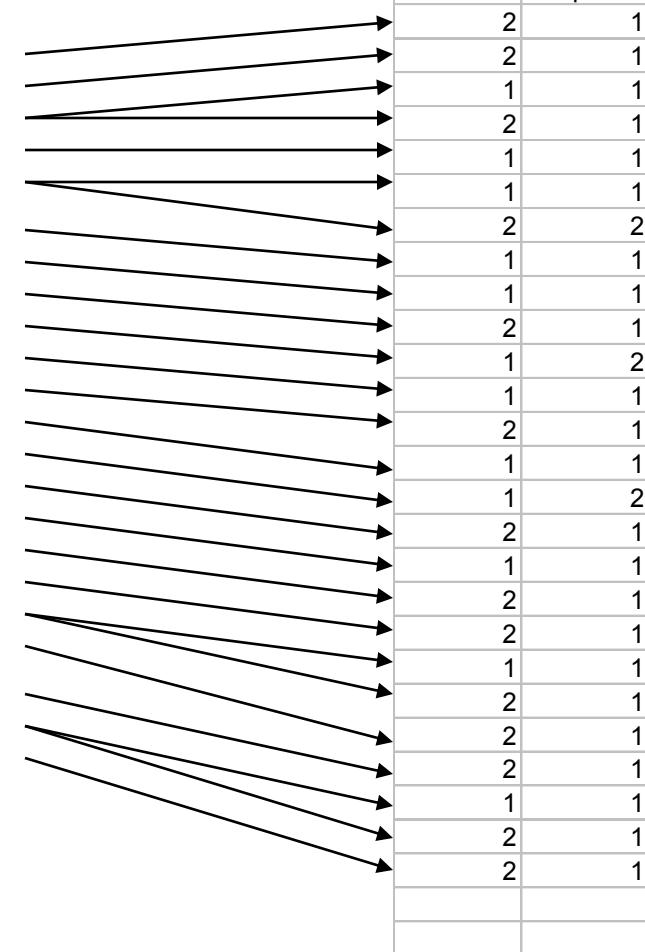
# Inverted index

Linked lists generally preferred to arrays

- Dynamic space allocation
- Insertion of terms into documents easy
- Space overhead of pointers
- In many practical applications Lucene is popular tool for indexing



# *Dictionary file and Postings file*

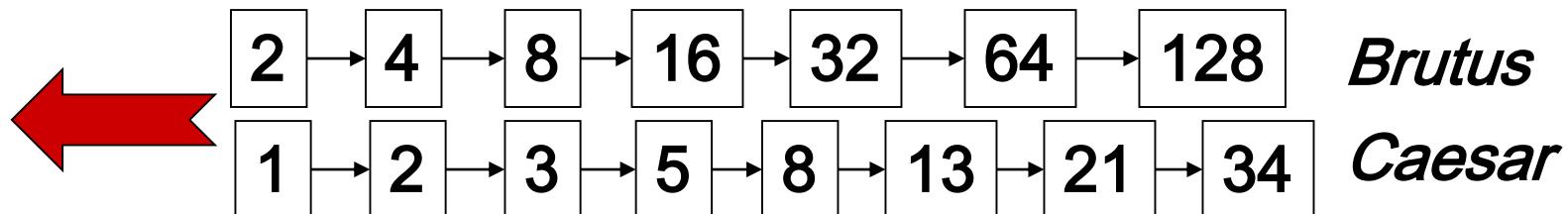


# Query processing: AND

Consider processing the query:

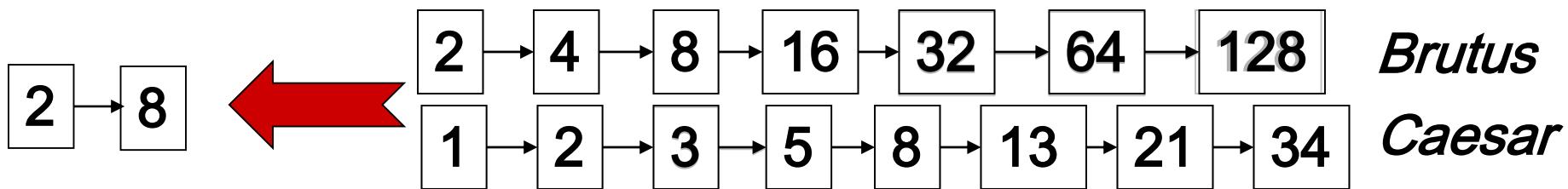
*Brutus AND Caesar*

- Locate *Brutus* in the Dictionary;
- Retrieve its postings.
- Locate *Caesar* in the Dictionary;
- Retrieve its postings.
- “Merge” the two postings:



# The merge

Walk through the two postings simultaneously, in time linear in the total number of postings entries



If the list lengths are  $x$  and  $y$ , the merge takes  $O(x+y)$  operations.

Crucial: postings sorted by docID.

# Merging Algorithm

---

Merge(p,q)

- 1 Start
2. Ans  $\leftarrow$  ()
3. While p  $\neq$  nil and q  $\neq$  nil do
  - if p  $\rightarrow$  docID = q  $\rightarrow$  docID  
then ADD(answer, p  $\rightarrow$  docID) // add to result and advance pointers
  - else if p  $\rightarrow$  docID < q  $\rightarrow$  docID  
then p  $\leftarrow$  p  $\rightarrow$  next
  - else q  $\leftarrow$  q  $\rightarrow$  next
4. end {of algo}

# Merging: More general merges

---

Consider an arbitrary Boolean formula:

*(Brutus OR Caesar) AND NOT*

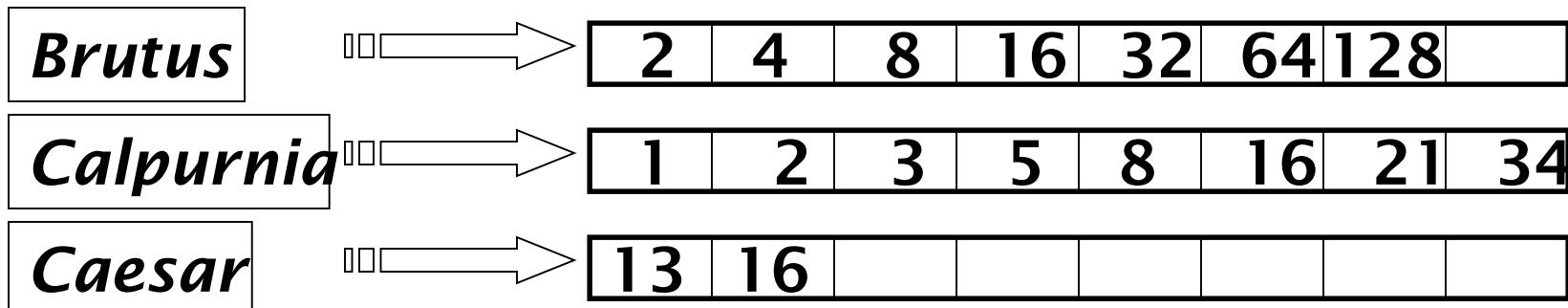
*(Antony OR Cleopatra)*

# Query optimization

What is the best order for query processing?

Consider a query that is an *AND* of  $t$  terms.

For each of the  $t$  terms, get its postings, then *AND* them together.



**Query:** *Brutus AND Calpurnia AND Caesar*

# Query Optimization

---

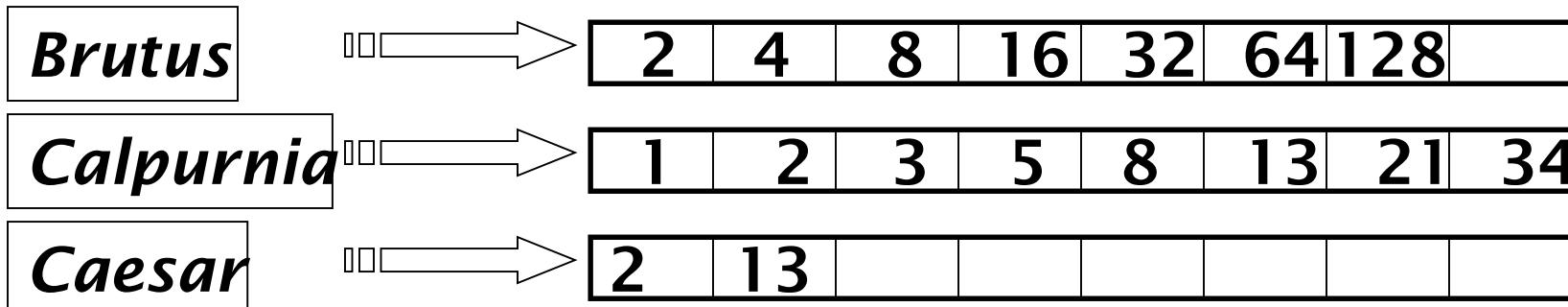
How to organize the work of getting results for a query so that the amount of work is reduced.

# Query optimization example

Process in order :

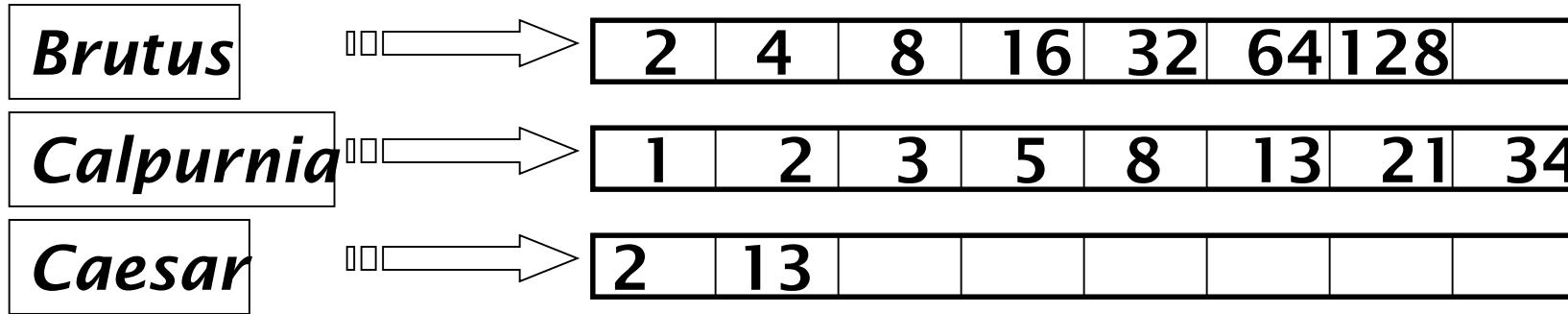
- start with smallest set of posting list, then keep cutting further.

This is why we kept  
order of documents



Execute the query as (*Caesar AND Brutus*) AND *Calpurnia*.

# Query optimization example



# **Brutus OR Caesar: 2, 4,8,13,16,32,64,128**

**(Brutus OR Caesar) AND NOT Calpurnia: 4, 16, 32, 64, 128**

**Execute the query *(Brutus OR Caesar) AND NOT Calpurnia*.**

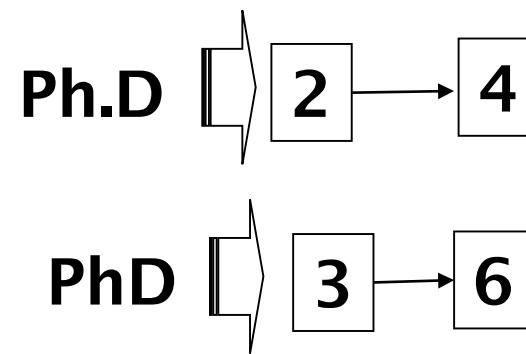
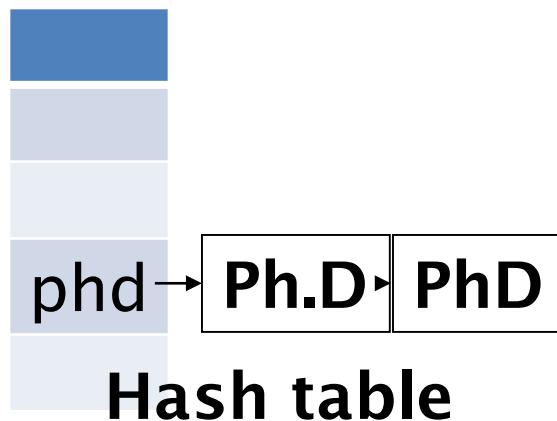
# Normalization

- The process of normalization is to **reduce multiple tokens to the same canonical term**, such that matches occur despite superficial differences.
- For example suppose you have variant forms of writing the same word like Ph.D, PhD which gets mapped to a single token as phd. Deleting the periods, hyphens, Accents etc..

# Query expansion / Asymmetric expansion

---

- For each term,  $t$ , in a query, expand the query with synonyms and related words of  $t$  from the thesaurus.
- Powerful but less efficient in terms of space
- These hand crafted terms is called as **Thesauri**.
- Handle Synonyms and Homonyms



# Stemming

- Stemming is the process of reducing inflectional form of words to their root form.
  - Example words like operation, operations, operational, operating can be reduced to operati (root word)
- Stemming is crude form of Normalization in which the suffixes are removed.
- The advantage of suffix stripping is to **reduce the total number of terms** in the inverted index resulting in a **smaller size and complexity of the data** in the system.

# Porter Stemmer

- A consonant in a word is a letter other than A, E, I, O or U
- Any letter not a consonant is a Vowel.
- All the words in English are of the form C(VC)<sup>m</sup>V where m is measure of any word or word part when represented in this form (VC).

**Examples:**

$m=0$  TR, EE, TREE, Y, BY.

$m=1$  TROUBLE, OATS, TREES

$m=2$  TROUBLES, PRIVATE, OATEN, ORRERY.

<https://snowballstem.org/algorithms/porter/stemmer.html>

# Porter Stemmer (contd..)

---

- The rules for removing a suffix will be given in the form (condition) S1 -> S2
- This means that if a word ends with the suffix S1, and the stem before S1 satisfies the given condition, S1 is replaced by S2.

(m > 1) EMENT -> E

Ex: Acknowledgement -> Acknowledge

---

# Porter Stemmer (contd..)

## Step 1a

SSES -> SS

caresses -> caress

IES -> I

ponies -> poni

SS -> SS

ties -> ti

S ->

caress -> caress

cats -> cat

- Step 1 deals with plurals and past participles.

## Step 1b

(m>0) EED -> EE

agreed -> agree

(\*v\*) ED ->

plastered -> plaster

bled -> bled

(\*v\*) ING ->

motoring -> motor

sing -> sing

- The subsequent steps are much more straightforward.

## Step 1c

(\*v\*) Y -> I

happy -> happi

sky -> sky

# Effect of stemming

- Suffix stripping of a vocabulary of 10,000 words

Number of words reduced in step 1: 3597

"	2:	766
"	3:	327
"	4:	2424
"	5:	1373

Number of words not reduced: 3650

- The resulting vocabulary of stems contained 6370 distinct entries.
- Thus the suffix stripping process reduced the size of the vocabulary by about one third.



# Variety of queries handled by search engine

- Boolean queries Ex: BITS and AIML
- Phrase queries Ex: “BITS AIML”
- Positional queries Ex: “BITS \3 AIML”
- Wild card queries Ex: a\*

# Phrase Queries

- Queries of the form “**BITS AIML**” where the word order needs to maintained.
- Two approaches
  - Biword Index
  - Positional Index

# Biword Index

- Index every consecutive pair of terms in the text as a phrase.
- For example “BITS AIML rocks”
  - **BITS AIML**
  - **AIML rocks**
- Disadvantage: False positives for longer queries

# Extended biwords

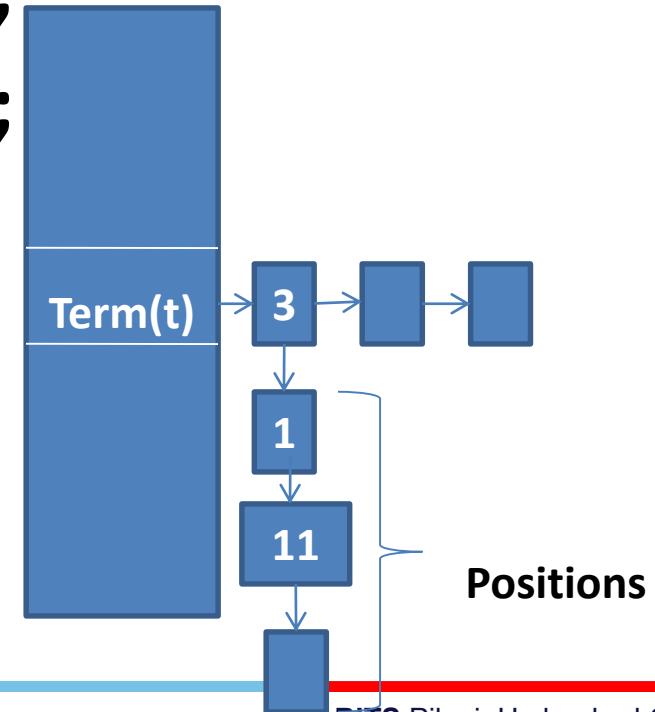
- 
- Parse the indexed text and perform **part-of-speech-tagging**.
  - Bucket the terms into (say) Nouns (N) and articles/prepositions (X).
  - Call any string of terms of the form  $NX^*N$  an **extended biword**.
    - Each such **extended biword** is now made a term in the dictionary.
  - Example: ***percyJackson and the torch***

N                    X    X    N

- Query processing: parse it into N's and X's
  - Segment query into enhanced biwords
  - Look up in index: ***percyJackson torch***

# Positional index

- In the postings, store for each *term* the position(s) in which tokens of it appear:  
*<term, number of docs containing term;*  
*doc1: position1, position2 ... ;*  
*doc2: position1, position2 ... ;*  
etc.>



# Example

Query = “The undiscovered country”

the:	undiscovered:	country:
3:34,38,55;	3:12,15,19;	3:22,26;
5:12,16,25,44;	5:3,5,17,41,45,96;	5:18,46,52,65;
7:67,87,90,101;	6:21,25,55,62;	7:5,69,91,105;
10:33,39,45,62;	7:4,68,70,85,110;	8:32,42,65,93;
	10:15,34,40,65,81;	10:32,44,75,83;

**The occurrence of “The undiscovered country”  
is found in the following:**

**Doc 5 (16,17,18) and (44,45,46)**

**Doc 7 (67,68,69)**

# Wildcard queries

- Trailing wildcard query
  - Ex: a\*
- Leading wildcard query
  - Ex: \*a

# Wildcard queries

- ca\* (e.g cab,cal,cam,can,cap,car,cat )
- Walk down the tree following c,a
  - Retrieve all words w such that:  $ca \leq w < cb$   
(i.e. all the words having prefix “ca”)
  - Let set of these terms be W.
  - Use inverted index to retrieve documents containing terms in W

# Trailing wildcard query

able
abstract
acute
baboon
baby
back
cab
cage
call
cam
dam
dance

[2-4] B-tree

[a-d]

[a-  
b]

[d]

[a]

[b]

[c]

a  
b  
l  
e

a  
b  
s  
t  
r  
a  
c  
t

a  
c  
u  
t  
e

b  
a  
b  
o  
o  
n

b  
a  
b  
y

b  
a  
c  
k

c  
a  
b

c  
a  
g  
e

c  
a  
l  
l

c  
a  
m

d  
a  
m

d  
a  
n  
c  
e

Trailing wildcard  
query:  $a^*$  ;  $a \leq w < b$

# Leading wildcard queries

aba  
aga  
aha  
baa  
bo  
a  
era  
eta  
goa  
kea  
kpa  
mq

aab  
aba  
abo  
ae  
aga  
aha  
aob  
aog  
apk  
aqm  
are  
ate

**[2-4] Reverse**

**B-tree**

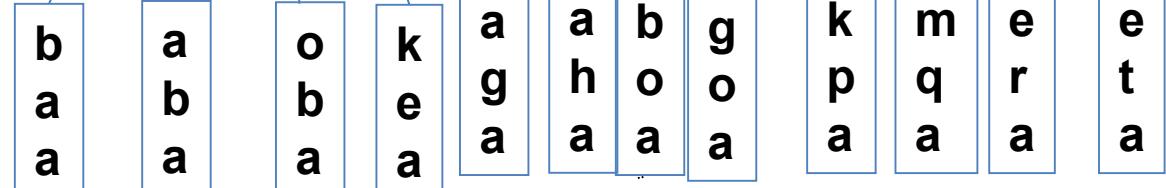
**[aa-ae]**

**[a]**

**[ag-ao]**

**[ap-ae]**

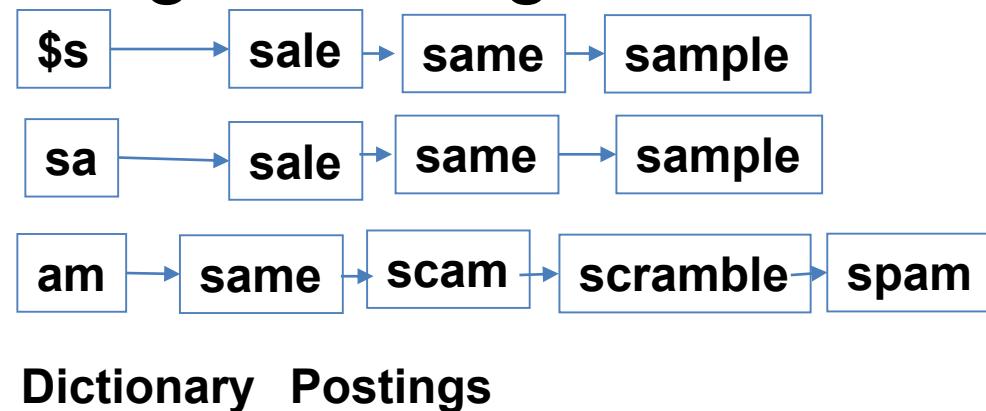
**Reverse the Terms and sort**



**Trailing wildcard query: \*oa ; ao ≤ w<ap**

# K-gram index

- Query: \$sam\*
- 2-grams from the query {\$s and sa and am}
- Fetch all the terms found in the posting list of the three K-grams using the merge algorithm



# Ranked Retrieval

- The user enters a **free text query** and we would like to **return the results in a ranked order**.
- Hence we need a **scoring scheme** to compute the relevance of the document to the user query.
- A simple scoring scheme in the range [0-1].

# Vector Space Model

---

Documents are represented as vectors in term space

- Terms are usually *stems*
- Documents represented by binary or weighted vectors of terms

Queries represented the same as documents

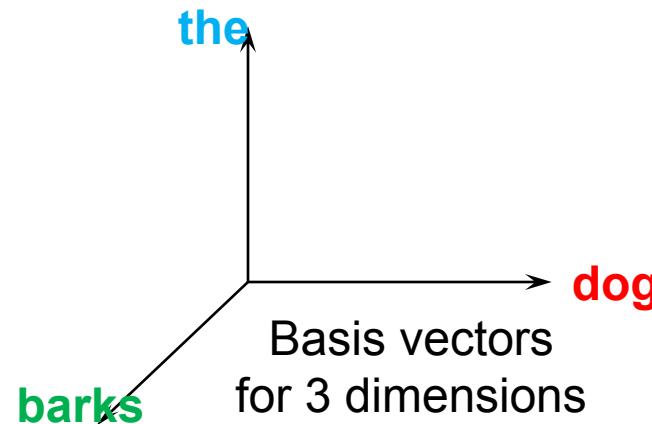
Query and Document weights are based on length and direction of their vector

A vector distance measure between the query and documents is used to rank retrieved documents

# Vector Space Representation from linear algebra perspective

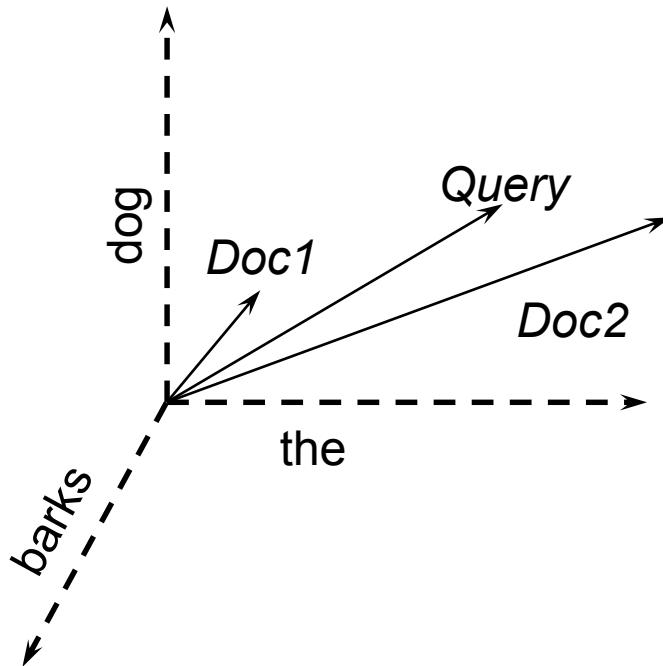


- Formally, a vector space is defined by a **set of linearly independent basis vectors**.
- Basis vectors:
  - correspond to the *dimensions* or *directions* in the vector space;
  - determine what can be described in the vector space; and
  - must be *orthogonal*, or *linearly independent*, i.e. a value along one dimension implies nothing about a value along another.



# Vector Coefficients

- How to represent the documents and queries?



Doc1: the dog barks <1 1 1>  
Doc2: the dog dog barks  
barks barks <1 2 3>

Query: the dog dog barks  
barks <1 2 2>

# Vector Space Similarity

Sim(X,Y)

Inner product  
(# nonzero dimensions)

Dice coefficient  
(Length normalized  
Inner Product)

Cosine coefficient  
(like Dice, but lower  
penalty with diff # features)

Jaccard coefficient  
(like Dice, but penalizes  
low overlap cases)

Binary Term Vectors

$$|X \cap Y|$$

$$\frac{2|X \cap Y|}{|X| + |Y|}$$

$$\frac{|X \cap Y|}{\sqrt{|X|} \sqrt{|Y|}}$$

$$\frac{|X \cap Y|}{|X| + |Y| - |X \cap Y|}$$

Weighted Term Vectors

$$\sum x_i \cdot y_i$$

$$\frac{2 \sum x_i y_i}{\sum x_i^2 + \sum y_i^2}$$

$$\frac{\sum x_i \cdot y_i}{\sqrt{\sum x_i^2} \cdot \sqrt{\sum y_i^2}}$$

$$\frac{\sum x_i \cdot y_i}{\sum x_i^2 + \sum y_i^2 - \sum x_i \cdot y_i}$$

# Jaccard coefficient: Example

- What is the query-document match score that the Jaccard coefficient computes for:
  - Query: “ides of March”
  - Document “Caesar died in March”
  - $\text{JACCARD}(q, d) = 1/6$

# What's wrong with Jaccard?

---

- It doesn't consider term frequency (how many occurrences a term has).
- Rare terms are more informative than frequent terms. Jaccard does not consider this information.
- Ex: Birla Institute of Technology and Science
- “Birla Institute” versus “of Technology”
- We need a more sophisticated way of normalizing for the length of a document  $|A \cap B| / \sqrt{|A \cup B|}$
- Later in this lecture, we'll use (cosine) instead of  $|A \cap B| / |A \cup B|$  (Jaccard) for length normalization.

# Binary incidence matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	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
...						

Each document is represented as a binary vector  $\in \{0, 1\}^{|V|}$ .

# Binary incidence matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	157	73	0	0	0	1
BRUTUS	4	157	0	2	0	0
CAESAR	232	227	0	2	1	0
CALPURNIA	0	10	0	0	0	0
CLEOPATRA	57	0	0	0	0	0
MERCY	2	0	3	8	5	8
WORSER	2	0	1	1	1	5
...						

Each document is now represented as a count vector

# Bag of words model

---

- We do not consider the **order** of words in a document.
- *John is quicker than Mary and Mary is quicker than John* are represented the same way.
- This is called a **bag of words model**.
- In a sense, this is a step back: The positional index was able to distinguish these two documents.
- For now: bag of words model

# Term frequency tf

---

- The term frequency  $tf_{t,d}$  of term  $t$  in document  $d$  is defined as the **number of times that  $t$  occurs in  $d$** .
- We want to use tf when computing query-document match scores.
- But how?
- Raw term frequency is not what we want because:
- A document with **tf = 10** occurrences of the term is more relevant than a document with **tf = 1** occurrence of the term.
- But not 10 times more relevant.
- Relevance does not increase proportionally with term frequency.

# Frequency in document vs. frequency in collection

---

- In addition, to term frequency (the frequency of the term in the document) . . .
- . . . we also want to use the frequency of the term **in the collection** for weighting and ranking.

# Desired weight for rare terms

---

- Rare terms are more informative than frequent terms.
- Consider a term in the query that is **rare** in the collection (e.g., ARACHNOCENTRIC).
- A document containing this term is very likely to be relevant.
- → We want **high weights for rare terms** like ARACHNOCENTRIC.

# Desired weight for frequent terms

- Frequent terms are less informative than rare terms.
- Consider a term in the query that is **frequent** in the collection (e.g., GOOD, INCREASE, LINE).
- A document containing this term is more likely to be relevant than a document that doesn't . . .
- . . . but words like GOOD, INCREASE and LINE are not sure indicators of relevance.
- → **For frequent terms** like GOOD, INCREASE and LINE, we want positive weights . . .
- . . . but **lower weights** than for rare terms.

# Document frequency

---

- We want **high weights** for rare terms like ARACHNOCENTRIC.
- We want **low (positive) weights** for frequent words like GOOD, INCREASE and LINE.
- We will use **document frequency** to factor this into computing the matching score.
- The document frequency is **the number of documents in the collection that the term occurs in**.

# idf weight

---

- $df_t$  is the document frequency, the number of documents that  $t$  occurs in.
- $df_t$  is an inverse measure of the informativeness of term  $t$ .
- We define the idf weight of term  $t$  as follows:

$$idf_t = \log_{10} \frac{N}{df_t}$$

( $N$  is the number of documents in the collection.)

- $idf_t$  is a measure of the informativeness of the term.
- $[\log N/df_t]$  instead of  $[N/df_t]$  to “dampen” the effect of idf

# Examples for idf

- Compute  $\text{idf}_t$  using the formula:

$$\text{idf}_t = \log_{10} \frac{1,000,000}{\text{df}_t}$$

term	$\text{df}_t$	$\text{idf}_t$
calpurnia	1	6
animal	100	4
sunday	1000	3
fly	10,000	2
under	100,000	1
the	1,000,000	0

# Effect of idf on ranking

---

- idf affects the ranking of documents for **queries with at least two terms**.
- For example, in the query “arachnocentric line”, idf weighting **increases** the relative weight of ARACHNOCENTRIC and **decreases** the relative weight of LINE.
- idf has **little effect on ranking for one-term queries**.

# tf-idf weighting

---

- The tf-idf weight of a term is the **product of its tf weight and its idf weight.**

$$w_{t,d} = (1 + \log \text{tf}_{t,d}) \cdot \log \frac{N}{\text{df}_t}$$

- tf-weight
- idf-weight
- Best known weighting scheme in information retrieval
- Note: the “-” in tf-idf is a hyphen, not a minus sign!
- Alternative names: tf.idf, tf x idf

# Summary: tf-idf

---

- Assign a tf-idf weight for each term  $t$  in each document  $d$ :

$$w_{t,d} = (1 + \log \text{tf}_{t,d}) \cdot \log \frac{N}{\text{df}_t}$$

- The tf-idf weight . . .
  - . . . increases with the number of occurrences within a document. (term frequency)
  - . . . increases with the rarity of the term in the collection. (inverse document frequency)

# Binary incidence matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	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
...						

Each document is represented as a binary vector  $\in \{0, 1\}^{|V|}$ .

# Count matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	157	73	0	0	0	1
BRUTUS	4	157	0	2	0	0
CAESAR	232	227	0	2	1	0
CALPURNIA	0	10	0	0	0	0
CLEOPATRA	57	0	0	0	0	0
MERCY	2	0	3	8	5	8
WORSER	2	0	1	1	1	5
...						

Each document is now represented as a count vector  $\in \mathbb{N}^{|V|}$ .

# Binary → count → weight matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	5.25	3.18	0.0	0.0	0.0	0.35
BRUTUS	1.21	6.10	0.0	1.0	0.0	0.0
CAESAR	8.59	2.54	0.0	1.51	0.25	0.0
CALPURNIA	0.0	1.54	0.0	0.0	0.0	0.0
CLEOPATRA	2.85	0.0	0.0	0.0	0.0	0.0
MERCY	1.51	0.0	1.90	0.12	5.25	0.88
WORSER	1.37	0.0	0.11	4.15	0.25	1.95
...						

Each document is now represented as a real-valued vector of tf idf weights  
 $\in \mathbb{R}^{|V|}$ .

# Documents as vectors

---

- Each document is now represented as a real-valued vector of tf-idf weights  $\in \mathbb{R}^{|V|}$ .
- So we have a  $|V|$ -dimensional real-valued vector space.
- Terms are **axes** of the space.
- Documents are **points** or **vectors** in this space.
- Very high-dimensional: tens of millions of dimensions when you apply this to web search engines
- Each vector is very sparse - most entries are zero.

# Cosine similarity between query and document

---

$$\cos(\vec{q}, \vec{d}) = \text{SIM}(\vec{q}, \vec{d}) = \frac{\vec{q} \cdot \vec{d}}{|\vec{q}| |\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2} \sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

- $q_i$  is the tf-idf weight of term  $i$  in the query.
- $d_i$  is the tf-idf weight of term  $i$  in the document.
- $|\vec{q}|$  and  $|\vec{d}|$  are the lengths of  $\vec{q}$  and  $\vec{d}$ .
- This is the **cosine similarity** of  $\vec{q}$  and  $\vec{d}$ . Or equivalently, the cosine of the angle between  $\vec{q}$  and  $\vec{d}$ .
- Larger cosine score (small angle), larger is similarity.

# Cosine for normalized vectors

---

- For normalized vectors, the cosine is equivalent to the dot product or scalar product.

$$\cos(\vec{q}, \vec{d}) = \vec{q} \cdot \vec{d} = \sum_i q_i \cdot d_i$$

- (if  $\vec{q}$  and  $\vec{d}$  are length-normalized).

# Cosine: Example

term frequencies (counts)

How similar are these novels?

SaS:

Sense and Sensibility

PaP:

Pride and Prejudice

WH:

Wuthering Heights

term	SaS	PaP	WH
AFFECTION	115	58	20
JEALOUS	10	7	11
GOSSIP	2	0	6
WUTHERING	0	0	38

# Cosine: Example

**term frequencies (counts)**

term	SaS	PaP	WH
AFFECTION	115	58	20
JEALOUS	10	7	11
GOSSIP	2	0	6
WUTHERING	0	0	38

**log frequency weighting**

term	SaS	PaP	WH
AFFECTION	3.06	2.76	2.30
JEALOUS	2.0	1.85	2.04
GOSSIP	1.30	0	1.78
WUTHERING	0	0	2.58

(To simplify this example, we don't do idf weighting.)

# Cosine: Example

log frequency weighting

term	SaS	PaP	WH
AFFECTION	3.06	2.76	2.30
JEALOUS	2.0	1.85	2.04
GOSSIP	1.30	0	1.78
WUTHERING	0	0	2.58

log frequency weighting & cosine normalization

term	SaS	PaP	WH
AFFECTION	0.789	0.832	0.524
JEALOUS	0.515	0.555	0.465
GOSSIP	0.335	0.0	0.405
WUTHERING	0.0	0.0	0.588

- $\cos(\text{SaS}, \text{PaP}) \approx$
- $0.789 * 0.832 + 0.515 * 0.555 + 0.335 * 0.0 + 0.0 * 0.0 \approx 0.94.$
- $\cos(\text{SaS}, \text{WH}) \approx 0.79$
- $\cos(\text{PaP}, \text{WH}) \approx 0.69$
- Why do we have  $\cos(\text{SaS}, \text{PaP}) > \cos(\text{SaS}, \text{WH})?$

$$\sqrt{(3.06)^2 + (2)^2 + (1.3)^2 + 0^2}$$

3.06

# Vector space model

---

- Represent the query as a weighted tf-idf vector
- Represent each document as a weighted tf-idf vector
- Compute the cosine similarity between the query vector and each document vector
- Rank documents with respect to the query
- Return the top  $K$  (e.g.,  $K = 10$ ) to the user

# References

---

<https://web.stanford.edu/class/cs276/handouts/lecture1-intro-handout-1-per.pdf>

IR Book

<https://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf>

Multimedia IR can be referred from

<http://people.ischool.berkeley.edu/~hearst/irbook/>

<https://towardsdatascience.com/stemming-lemmatization-what-ba782b7c0bd8>

<https://www.analyticsvidhya.com/blog/2021/11/an-introduction-to-stemming-in-natural-language-processing/>

<https://snowballstem.org/algorithms/porter/stemmer.html>

# Agenda

---

- Parts Of Speech tagging (POS)
  - PennTree dataset
  - Measuring performance of POS taggers
  - Two approaches to POS tagging
  - Hidden Markov Model
  - Viterbi Algorithm
-

# POS Tagging

---

The process of assigning a part-of-speech or lexical class marker to each word in a sentence (and all sentences in a collection).

Input: the lead paint is unsafe

Output: the/Det lead/N paint/N is/V unsafe/Adj

Don't worry! There is no problem with your eyes or computer.

# Let's try

ଓ/DT এৰেল/NN ০ ১০/VBZ কোও/DT ০ ৫ল/VBG ও/DT  
কো/NN পত্র/.

ଓ/DT এৰে ৪/NN ০ ১০/VBZ ৯ ১ ৫ ৫ ০ ৫ল/VBG পত্র/.

ଓ/DT কু ৬ ৫/NN ০ ১০/VBZ ১০ ০ ৫ল ০ ৫ল/VBG পত্র/.

ଓ/DT কো ৭ ৭ ৫/JJ কু ০ ৯ল/NN

What is the POS tag sequence of the following sentence?

ଓ কো ৭ ৭ ৫ কো ৩কো ১০ ১০ ০ ৫ল ০ ৫ল পত্র

# Let's try

---

କୁ/DT କୁଥେ/NN ୦ ୧୦/VBZ ହୋଇଥିଲେ/VBG କୁ/DT      ହୋଇଥିଲେ/VBG /.

a/DT dog/NN is/VBZ chasing/VBG a/DT cat/NN ./.

କୁ/DT କୁଥେ/NN ୦ ୧୦/VBZ ୨ ୧ ୫ ୫ ୦ ୫ାଲେ/VBG /.

a/DT fox/NN is/VBZ running/VBG ./.

କୁ/DT କୁଥେ/NN ୦ ୧୦/VBZ ୧୦ ୦ ୫ାଲେ/VBG /.

a/DT boy/NN is/VBZ singing/VBG ./.

କୁ/DT କୁଥେ/NN ୦ ୧୦/VBZ ୧୦ ୦ ୫ାଲେ/VBG /.

a/DT happy/JJ bird/NN

କୁ/DT କୁଥେ/NN ୦ ୧୦/VBZ ୧୦ ୦ ୫ାଲେ/VBG /.

a happy cat was singing .

# Why is POS Tagging Useful?

---

- First step of a vast number of practical tasks
- Parsing
  - Need to know if a word is an N or V before you can parse
- Word sense disambiguation
- Information Extraction
  - Finding names, relations, etc.
- Machine Translation
- Named Entity Recognition (NER)
- Sentiment analysis
- Question answering/Chat bots

# Parts of Speech

---

8 (ish) traditional parts of speech

- Noun, verb, adjective, preposition, adverb, article, interjection, pronoun, conjunction, etc
- Called: parts-of-speech, lexical categories, word classes, morphological classes, lexical tags...

# POS examples

---

N	noun	<i>chair, bandwidth, pacing</i>
V	verb	<i>study, debate, munch</i>
ADJ	adjective	<i>purple, tall, ridiculous</i>
ADV	adverb	<i>unfortunately, slowly</i>
P	preposition	<i>of, by, to</i>
PRO	pronoun	<i>I, me, mine</i>
DET	determiner	<i>the, a, that, those</i>

# POS Tagging

---

The process of assigning a part-of-speech or lexical class marker to each word in a collection.

<u>WORD</u>	<u>tag</u>
the	DET
koala	N
put	V
the	DET
keys	N
on	P
the	DET
table	N

# Open and Closed Classes

---

Closed class: a small fixed membership

- Prepositions: of, in, by, ...
- Auxiliaries: may, can, will had, been, ...
- Pronouns: I, you, she, mine, his, them, ...
- Usually **function words** (short common words which play a role in grammar)

Open class: new ones can be created all the time

- English has 4: Nouns, Verbs, Adjectives, Adverbs
- Many languages have these 4, but not all!

# Open Class Words

## Nouns

- Proper nouns (Boulder, Granby, Eli Manning)
  - English capitalizes these.
- Common nouns (the rest).
- Count nouns and mass nouns
  - Count: have plurals, get counted: goat/goats, one goat, two goats
  - Mass: don't get counted (snow, salt, communism) (\*two snows)

Adverbs: tend to modify things

- **Unfortunately**, John walked home **extremely slowly yesterday**
- Directional/locative adverbs (here, home, downhill)
- Degree adverbs (extremely, very, somewhat)
- Manner adverbs (slowly, slinkily, delicately)

Verbs

- In English, have morphological affixes (eat/eats/eaten)

# Closed Class Words

---

## Examples:

- prepositions: *on, under, over, ...*
- particles: *up, down, on, off, ...*
- determiners: *a, an, the, ...*
- pronouns: *she, who, I, ..*
- conjunctions: *and, but, or, ...*
- auxiliary verbs: *can, may should, ...*
- numerals: *one, two, three, third, ...*

# Ambiguities in language

---

## I. Structural Ambiguities

- Visiting relatives can be nuisance. (two meanings)

## II. Grammatical Ambiguities

- I (feminine or masculine) go.
- Can- Noun = container, Can – Modal(auxiliary verb),  
Can-verb = to can means to pack etc

## III. Lexical Ambiguities:

Polysemy Ex: "understand" (I get it)

- Homonymy Ex: Bank= river, financial bank

# What is the challenge in PoS Tagging?



- Tag ambiguous words
  - Solve the lexical ambiguities
  - The/DT wind/NN was/VB too/ADV strong/ADJ to/PRP  
wind/VB the/DT sail/NN.
- Tag unknown words
  - The/DT rural/JJ Babbitt/??? who/WP bloviates/???  
about/IN progress/NN and/CC growth/NN

# POS Tagging

## Choosing a Tagset

---

- There are so many parts of speech, potential distinctions we can draw
- To do POS tagging, we need to choose a standard set of tags to work with
- Could pick very coarse tagsets
  - N, V, Adj, Adv.
- More commonly used set is finer grained, the “Penn TreeBank tagset”, 45 tags
  - PRP\$, WRB, WP\$, VBG
- Even more fine-grained tagsets exist

# Penn TreeBank POS Tagset

Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	<i>and, but, or</i>	SYM	symbol	+%, &
CD	cardinal number	<i>one, two, three</i>	TO	“to”	<i>to</i>
DT	determiner	<i>a, the</i>	UH	interjection	<i>ah, oops</i>
EX	existential ‘there’	<i>there</i>	VB	verb, base form	<i>eat</i>
FW	foreign word	<i>mea culpa</i>	VBD	verb, past tense	<i>ate</i>
IN	preposition/sub-conj	<i>of, in, by</i>	VBG	verb, gerund	<i>eating</i>
JJ	adjective	<i>yellow</i>	VBN	verb, past participle	<i>eaten</i>
JJR	adj., comparative	<i>bigger</i>	VBP	verb, non-3sg pres	<i>eat</i>
JJS	adj., superlative	<i>wildest</i>	VBZ	verb, 3sg pres	<i>eats</i>
LS	list item marker	<i>1, 2, One</i>	WDT	wh-determiner	<i>which, that</i>
MD	modal	<i>can, should</i>	WP	wh-pronoun	<i>what, who</i>
NN	noun, sing. or mass	<i>llama</i>	WP\$	possessive wh-	<i>whose</i>
NNS	noun, plural	<i>llamas</i>	WRB	wh-adverb	<i>how, where</i>
NNP	proper noun, singular	<i>IBM</i>	\$	dollar sign	\$
NNPS	proper noun, plural	<i>Carolinas</i>	#	pound sign	#
PDT	predeterminer	<i>all, both</i>	“	left quote	‘ or “
POS	possessive ending	<i>'s</i>	”	right quote	’ or ”
PRP	personal pronoun	<i>I, you, he</i>	(	left parenthesis	[, (, {, <
PRP\$	possessive pronoun	<i>your, one's</i>	)	right parenthesis	], ), }, >
RB	adverb	<i>quickly, never</i>	,	comma	,
RBR	adverb, comparative	<i>faster</i>	.	sentence-final punc	. ! ?
RBS	adverb, superlative	<i>fastest</i>	:	mid-sentence punc	: ; ... – -
RP	particle	<i>up, off</i>			

# Using the Penn Tagset

---

- The/DT grand/JJ jury/NN commented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS ./.
- Prepositions and subordinating conjunctions marked IN (“although/IN I/PRP..”)
- Except the preposition “to” is just marked “TO”.

# POS Tagging

---

- Words often have more than one POS: *back*
  - The **back** door = JJ
  - On my **back** = NN
  - Win the voters **back** = RB
  - Promised to **back** the bill = VB
- The POS tagging problem is to determine the POS tag for a particular instance of a word.

# POS Tagging performance evaluation



- Percentage of tags predicted correctly.
- Baseline approach: Tag every word with its most common tag and rest of the words are tagged as Noun.



# Two Methods for PoS Tagging

- Rule-based systems
- Statistical sequence models
- Hidden Markov Models

# POS Tagging as Sequence Classification

---

- We are given a sentence (an “observation” or “sequence of observations”)
  - *Secretariat is expected to race tomorrow*
- What is the best sequence of tags that corresponds to this sequence of observations?
- Probabilistic view
  - Consider all possible sequences of tags
  - Out of this universe of sequences, choose the tag sequence which is most probable given the observation sequence of n words  $w_1 \dots w_n$ .

# Information Extraction

---

- Identify phrases in language that refer to specific types of entities and relations in text.
- Named entity recognition is task of identifying names of people, places, organizations, etc. in text.
- **people    organizations    places**
  - Michael Dell is the CEO of Dell Computer Corporation and lives in Austin Texas.
- Extract pieces of information relevant to a specific application, e.g. used car ads:
- **make    model    year    mileage    price**
  - For sale, 2002 Toyota Prius, 20,000 mi, \$15K or best offer. Available starting July 30, 2006.

# Bioinformatics

---

- Sequence labeling also valuable in labeling genetic sequences in genome analysis.

extron intron

– AGCTAACGTTCGATACG**GATTACA**GCCT

# Language Modelling

- The task of a language model is to **express the restrictions imposed** on the way in which words can be combined to form sentences.

# Stochastic Language Model

- 
- The task of a stochastic language model is to provide estimates of the prior probability of the sentence generation using the training sentences.

# Jargons

- Corpus: - set of training sentences.
- Vocabulary: finite set of words used in writing sentences.
  - For example  $V = \{\text{the}, \text{dog}, \text{barks}, \text{cat}, \text{smiles}, \text{boy}, \text{play}..\}$
- $V'$  = Set of all possible sentences in this language.

the dog barks STOP(well formed sentence)

the cat cat STOP(ill formed sentence)

the the the STOP(ill formed sentence)

the cat smiles STOP(well formed sentence)

STOP

# Stochastic Language Model (Contd..)

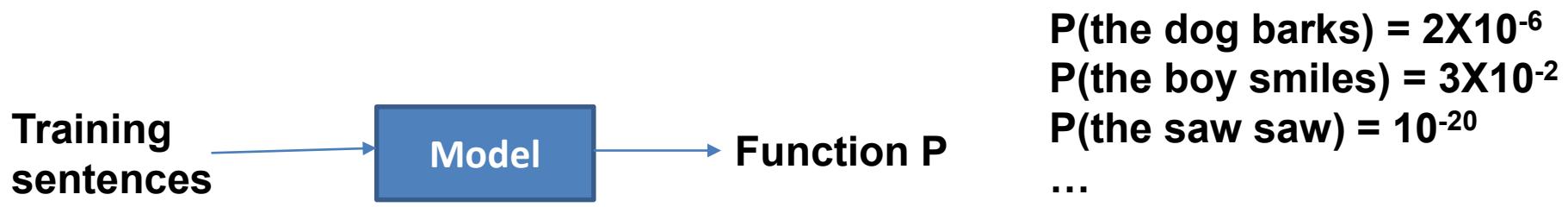
---



- Assume that we have a training sentences in English.
  - These could be easily collected from online newspapers or webpages.
  - Given these training samples the task of Language Model is learn a distribution  $P$  over sentences in our language.
  - $P$  is going to be a function which must satisfy the following two constraints

# Stochastic Language Model

- We would prefer a good language model to assign high probability to a sentence which occurs in English.



- The task of function P is to assign probability to every sentence in the language.

# A naïve language model



- Given  $N$  training sentences, the task is to **learn** a distribution  $P$  over sentences in the language.
- For any sentence  $w_1, w_2, \dots, w_n$  let  $c(w_1, w_2, \dots, w_n)$  denote the number of times this sentence is seen in the corpus.

# Generative model for sequence labelling

---



- Given a set of training examples  $\langle x_i, y_i \rangle$  for  $i=1..m$  where each  $x_i$  is the input vector and  $y_i$  is the class label.
- In the POS tagging problem we have
  - $x_1 = \text{the dog barks}, y_1 = \text{DT NN VB}$
  - $x_2 = \text{the boy smiles}, y_2 = \text{DT NN VB}$
  - .....
- The task is to learn a function  $f$  that maps input  $x$  to its corresponding labels  $f(x)$ .

# Generative model for sequence labelling



Using Bayes rule

$$P(t_1, \dots, t_n | w_1, \dots, w_n) = \frac{P(t_1, \dots, t_n)P(w_1, \dots, w_n | t_1, \dots, t_n)}{P(w_1, \dots, w_n)}$$

Submodels:

1. Prior:  $P(t_1, \dots, t_n)$
2. Likelihood:  $P(w_1, \dots, w_n | t_1, \dots, t_n)$
3. Marginal:  $P(w_1, \dots, w_n)$  – can be ignored in argmax search

# Statistical POS Tagging

We want, out of all sequences of n tags  $t_1 \dots t_n$ , the single tag sequence such that

$$P(t_1 \dots t_n | w_1 \dots w_n) \text{ is highest.}$$

$$\hat{t}_1^n = \operatorname{argmax}_{t_1^n} P(t_1^n | w_1^n)$$

Hat ^ means “our estimate of the best one”

Argmax<sub>x</sub> f(x) means “the x such that f(x) is maximized”

# Statistical POS Tagging

This equation should give us the best tag sequence

$$\hat{t}_1^n = \operatorname{argmax}_{t_1^n} P(t_1^n | w_1^n)$$

But how to make it operational? How to compute this value?

Intuition of Bayesian inference:

- Use Bayes rule to transform this equation into a set of probabilities that are easier to compute (and give the right answer)

# Using Bayes Rule

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$

$$\hat{t}_1^n = \operatorname{argmax}_{t_1^n} P(t_1^n | w_1^n)$$

$$\hat{t}_1^n = \operatorname{argmax}_{t_1^n} \frac{P(w_1^n | t_1^n)P(t_1^n)}{P(w_1^n)}$$

$$\hat{t}_1^n = \operatorname{argmax}_{t_1^n} P(w_1^n | t_1^n)P(t_1^n)$$

# How you predict the tags?

---

Two types of information are useful

- Relations between words and tags
- Relations between tags and tags
  - DT NN, DT JJ NN...

Secretariat/**NNP** is/**VBZ** expected/**VBN** to/**TO** race/**VB** tomorrow/**NR**

People/**NNS** continue/**VB** to/**TO** inquire/**VB** the/**DT** reason/**NN** for/**IN** the/**DT** race/**NN** for/**IN** outer/**JJ** space/**NN**

# Statistical POS tagging

What is the most likely sequence of tags for the given sequence of words  $w$

$$\begin{aligned}
 \operatorname{argmax}_{\mathbf{t}} P(\mathbf{t}|\mathbf{w}) &= \operatorname{argmax}_{\mathbf{t}} \frac{P(\mathbf{t}, \mathbf{w})}{P(\mathbf{w})} \\
 &= \operatorname{argmax}_{\mathbf{t}} P(\mathbf{t}, \mathbf{w}) \\
 &= \operatorname{argmax}_{\mathbf{t}} P(\mathbf{t})P(\mathbf{w}|\mathbf{t})
 \end{aligned}$$

 $P(\text{ DT JJ NN } | \text{ a smart dog})$   
 $= P(\text{DD JJ NN a smart dog}) / P(\text{a smart dog})$

# Markov Chains

- A Markov chain is a model that tells us something about the **probabilities of sequences of random variables**, states, each of which can take on values from some set.
- A Markov Model is a finite state machine with probabilistic state transitions.
- Markov assumption that **next state only depends on the current state** and independent of previous history.

# Transition Probability

---

- Joint probability  $P(\mathbf{t}, \mathbf{w}) = P(\mathbf{t})P(\mathbf{w}|\mathbf{t})$
- $P(\mathbf{t}) = P(t_1, t_2, \dots, t_n)$ 

$$= P(t_1)P(t_2 | t_1)P(t_3 | t_2, t_1) \dots P(t_n | t_1 \dots t_{n-1})$$

$$\sim P(t_1)P(t_2 | t_1)P(t_3 | t_2) \dots P(t_n | t_{n-1})$$

$$= \prod_{i=1}^n P(t_i | t_{i-1})$$

**Markov assumption**

- Bigram model over POS tags!  
(similarly, we can define a n-gram model over POS tags, usually we called high-order HMM)

# Emission Probability

---

- Joint probability  $P(\mathbf{t}, \mathbf{w}) = P(\mathbf{t})P(\mathbf{w}|\mathbf{t})$
- Assume words only depend on their POS-tag
- $P(\mathbf{w}|\mathbf{t}) \sim P(w_1 | t_1)P(w_2 | t_2) \dots P(w_n | t_n)$

$$= \prod_{i=1}^n P(w_i | t_i)$$

Independent assumption

i.e.,  $P(\text{a smart dog} | \text{DD JJ NN})$

$$= P(\text{a} | \text{DD}) P(\text{smart} | \text{JJ}) P(\text{dog} | \text{NN})$$

# Put them together

---

- Joint probability  $P(\mathbf{t}, \mathbf{w}) = P(\mathbf{t})P(\mathbf{w}|\mathbf{t})$
  - $P(\mathbf{t}, \mathbf{w})$   
 $= P(t_1)P(t_2 | t_1)P(t_3 | t_2) \dots P(t_n | t_{n-1})$   
 $P(w_1 | t_1)P(w_2 | t_2) \dots P(w_n | t_n)$   
 $= \prod_{i=1}^n P(w_i | t_i)P(t_i | t_{i-1})$
- e.g.,  $P(\text{a smart dog , DD JJ NN })$   
 $= P(\text{a} | \text{DD}) P(\text{smart} | \text{JJ }) P(\text{dog} | \text{NN })$   
 $P(\text{DD} | \text{start}) P(\text{JJ} | \text{DD}) P(\text{NN} | \text{JJ })$

# Likelihood and Prior



$$\hat{t}_1^n = \operatorname{argmax}_{t_1^n} \frac{\overbrace{P(w_1^n | t_1^n)}^{\text{likelihood}}}{\overbrace{P(t_1^n)}^{\text{prior}}}$$

$$P(w_1^n | t_1^n) \approx \prod_{i=1}^n P(w_i | t_i)$$



$$P(t_1^n) \approx \prod_{i=1}^n P(t_i | t_{i-1})$$

$$\hat{t}_1^n = \operatorname{argmax}_{t_1^n} P(t_1^n | w_1^n) \approx \operatorname{argmax}_{t_1^n} \prod_{i=1}^n P(w_i | t_i) P(t_i | t_{i-1})$$

# Two Kinds of Probabilities

---

## 1. State transition probabilities -- $p(t_i|t_{i-1})$

- State-to-state transition probabilities

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

## 2. Observation/Emission probabilities -- $p(w_i|t_i)$

- Probabilities of observing various values at a given state

$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

# Two Kinds of Probabilities

## 1. Tag transition probabilities -- $p(t_i|t_{i-1})$

- Determiners likely to precede adjs and nouns
  - That/DT flight/NN
  - The/DT yellow/JJ hat/NN
  - So we expect  $P(NN|DT)$  and  $P(JJ|DT)$  to be high
- Compute  $P(NN|DT)$  by counting in a labeled corpus:

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

$$P(NN|DT) = \frac{C(DT, NN)}{C(DT)} = \frac{56,509}{116,454} = .49$$

# Two Kinds of Probabilities

## 2. Word likelihood/emission probabilities

$$p(w_i|t_i)$$

- VBZ (3sg Pres Verb) likely to be “is”
- Compute  $P(is|VBZ)$  by counting in a labeled corpus:

$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

$$P(is|VBZ) = \frac{C(VBZ, is)}{C(VBZ)} = \frac{10,073}{21,627} = .47$$

# Sample Transition Probabilities

## *Bigram Probabilities*

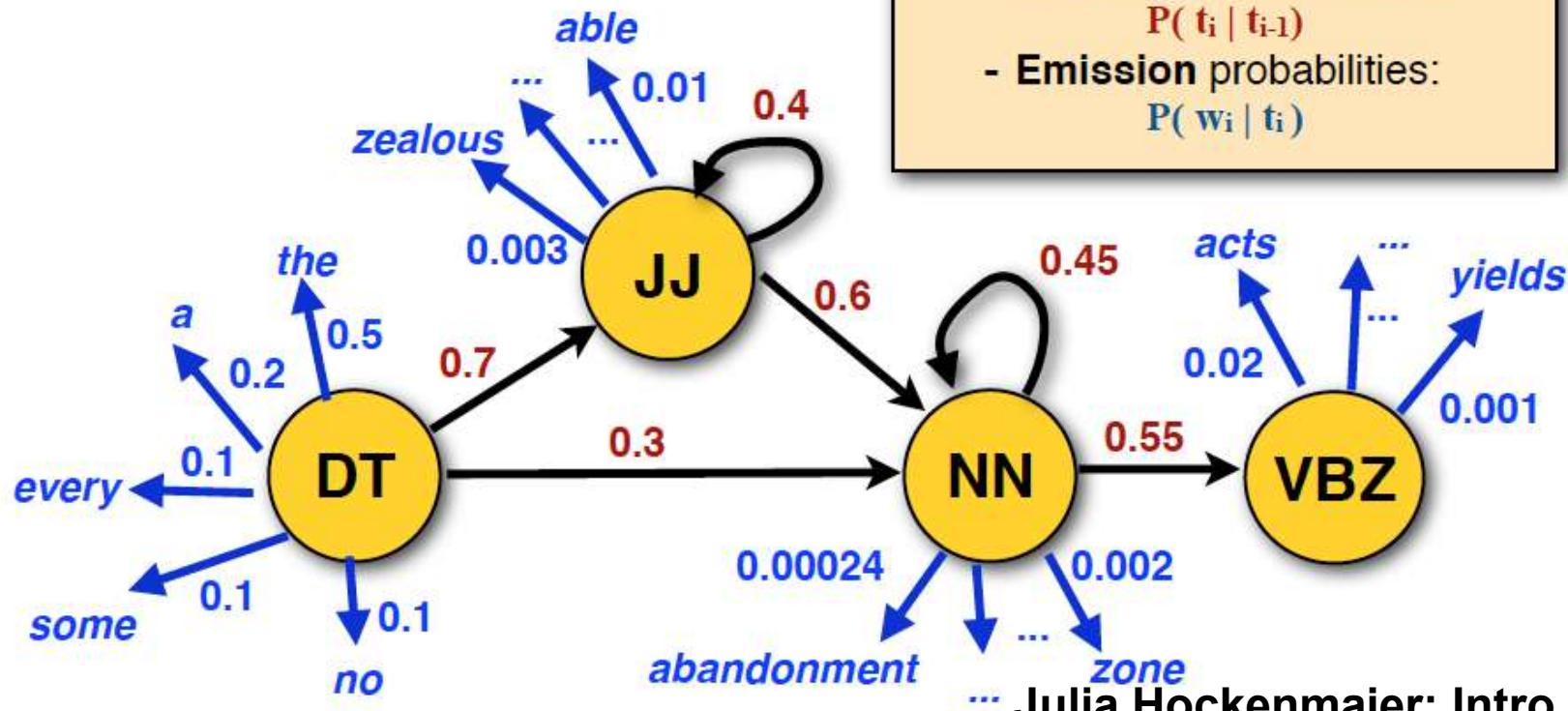
• Bigram(Ti, Tj)	Count(i, i + 1)	Prob(Tj Ti)
• φ,ART	213	.71 (213/300)
• φ,N	87	.29 (87/300)
• φ,V	10	.03 (10/300)
• ART,N	633	1
• N,V	358	.32
• N,N	108	.10
• N,P	366	.33
• V,N	134	.37
• V,P	150	.42
• V,ART	194	.54
• P,ART	226	.62
• P,N	140	.38
• V,V	30	.08

## *Tag Frequencies*

Φ	ART	N	V	P
300	633	1102	358	366

Φ – Start Symbol

# HMMs as probabilistic FSA



# Markov Assumption

---

- Context model (prior)

$$P(t_1, \dots, t_n) = \prod_{i=1}^n P(t_i | t_{i-k}, \dots, t_{i-1})$$

- Lexical model (likelihood)

$$P(w_1, \dots, w_n | t_1, \dots, t_n) = \prod_{i=1}^n P(w_i | t_i)$$

# Model Parameters

- Contextual probabilities :  $P(t_i|t_{i-k}, \dots, t_{i-1})$
- Lexical probabilities :  $P(w_i|t_i)$
- We can estimate these probabilities from a tagged corpus:

$$\hat{P}_{MLE}(w_i|t_i) = \frac{c(w_i, t_i)}{c(t_i)} \quad \hat{P}_{MLE}(t_i|t_{i-k}, \dots, t_{i-1}) = \frac{c(t_{i-k}, \dots, t_{i-1}, t_i)}{c(t_{i-k}, \dots, t_{i-1})}$$

# Computing Probabilities

- The probability of a tagging:

$$P(t_1, \dots, t_n, w_1, \dots, w_n) = \prod_{i=1}^n P(t_i | t_{i-k}, \dots, t_{i-1}) P(w_i | t_i)$$

- Finding the most probable tagging:

$$\operatorname{argmax}_{t_1, \dots, t_n} \prod_{i=1}^n P(t_i | t_{i-k}, \dots, t_{i-1}) P(w_i | t_i)$$

# Example

- Given a sentence of length 3, \* \* the dog barks STOP and the tag sequence \* \* DT NN VB \* then
- $P(w_1 w_2 w_3, y_1, y_2, y_3) = T(DT|*, *) \times T(NN|*, DT) \times T(VB|DT\ NN) \times T(STOP|NN\ VB) \times E(*|*) \times E(*|*) E(the|DT) \times E(dog|NN) \times E(barks|VB) \times E(STOP|*)$
- We can also define  $y_{-1} = ^*$  and  $y_0 = ^*$  as special symbols.

# Hidden Markov Models (formal)

---

States  $T = t_1, t_2 \dots t_N$ ;

Observations  $W = w_1, w_2 \dots w_N$ ;

- Each observation is a symbol from a vocabulary  $V = \{v_1, v_2, \dots, v_V\}$

## Transition probabilities

- Transition probability matrix  $A = \{a_{ij}\}$

$$a_{ij} = P(t_i = j \mid t_{i-1} = i) \quad 1 \leq i, j \leq N$$

## Observation likelihoods

- Output probability matrix  $B = \{b_i(k)\}$

$$b_i(k) = P(w_i = v_k \mid t_i = i)$$

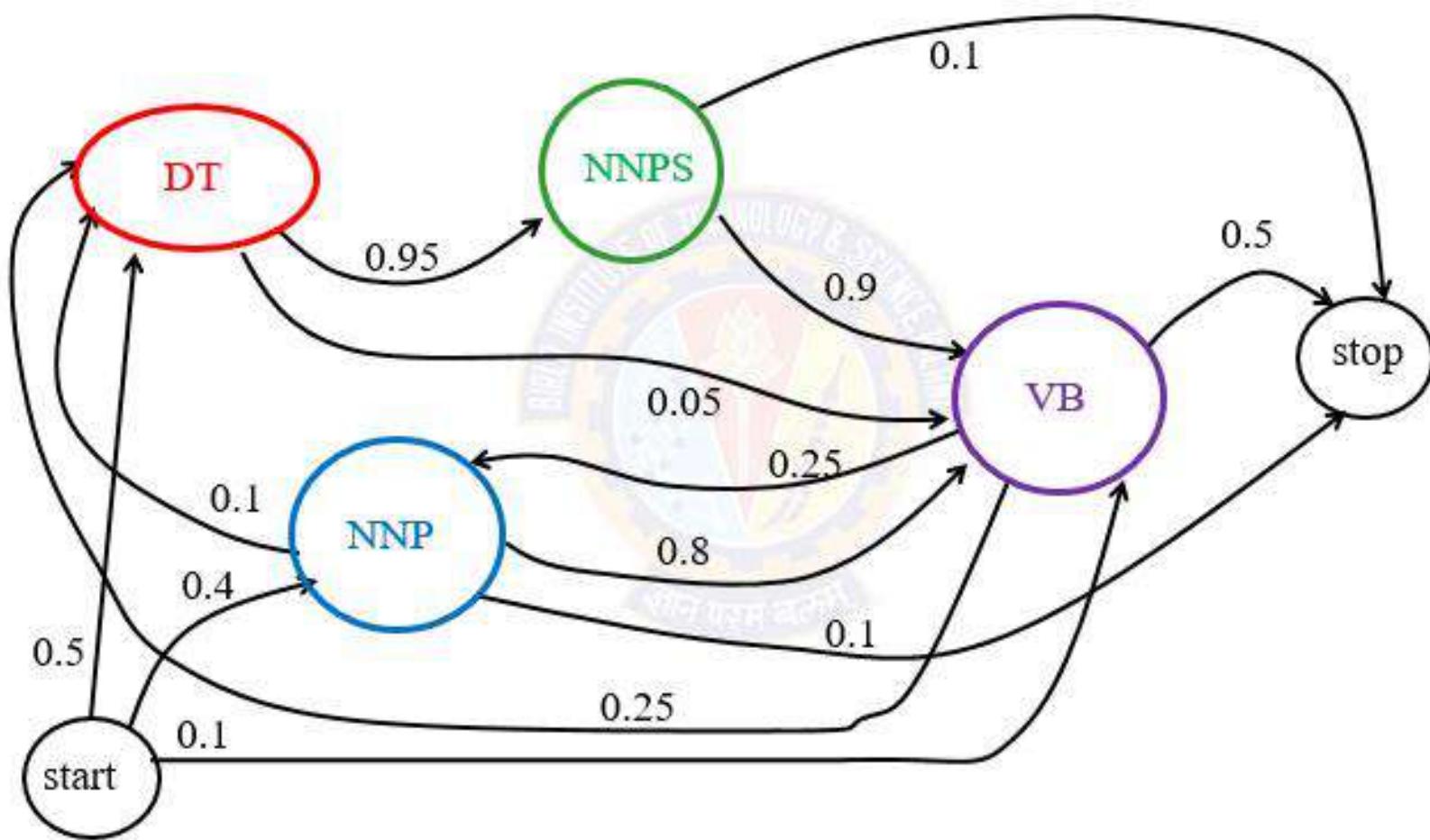
## Special initial probability vector $\pi$

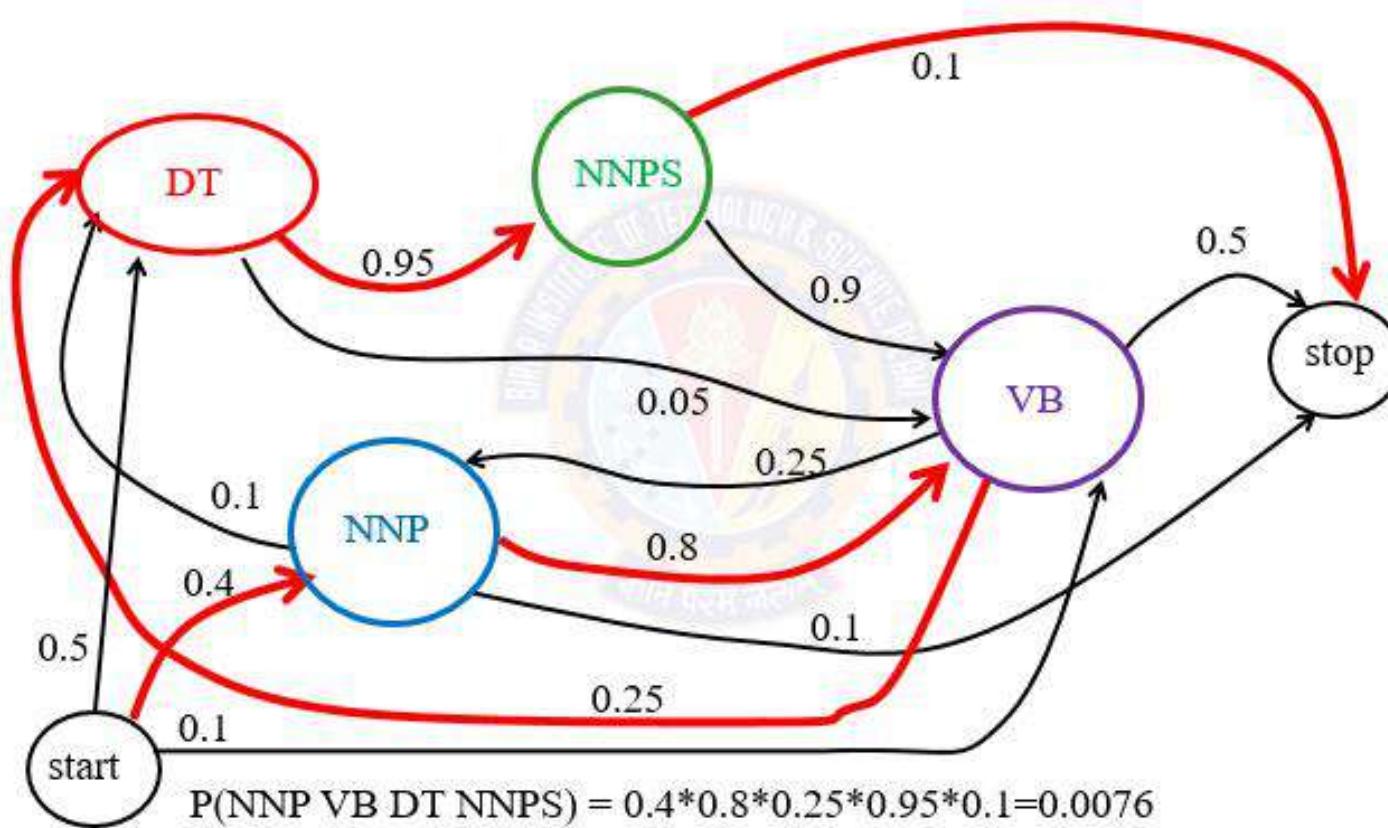
$$\pi_i = P(t_1 = i) \quad 1 \leq i \leq N$$

# Markov Chain

- Formally, a Markov chain is specified by the following components:

$Q = q_1 q_2 \dots q_N$ a set of N states	A set of N states
$A = a_{11} a_{12} \dots a_{n1} \dots a_{nn}$	A transition probability matrix A, each $a_{ij}$ representing the probability of moving from state i to state j,
$\pi = \pi_1, \pi_2, \pi_3, \dots, \pi_n$	An initial probability distribution over states. $\pi_i$ is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$ , meaning that they cannot be initial states.





# Example

Emission Matrix

	*	DT	NN S	VB	NN	IN	STOP
*	1						
the		3/4					
employees			3/4				
pass				2/4			
an		1/4					
exam					1		
wait				1/4			
for						1	
employers			1/4				
fire				1/4			
.							1

Tag Translation Matrix

SECOND TAG

F R I S T	*	DT	NNS	VB	NN	IN	STOP
T A G	*	2/3	1/3				
DT			2/4	1/4	1/4		
NNS				3/4			1/4
VB		1/4	1/4			1/4	1/4
NN							1
IN		1					
STOP							

S1: the Employees pass an exam .

T1: DT NNS                      VB    DT    NN    STOP

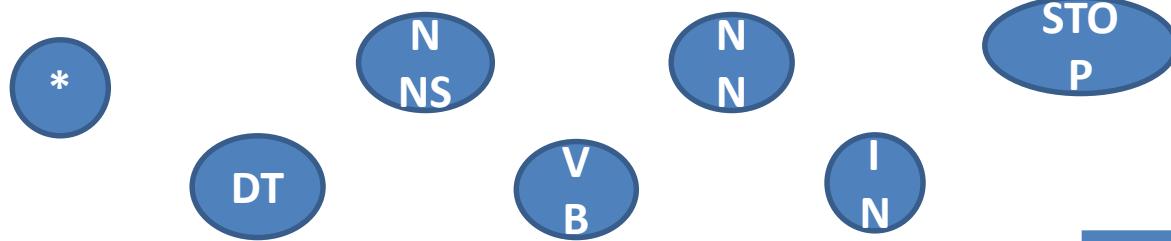
S2: the employees wait for the pass .

T2: DT            NNS            VB    IN    DT    VB    STOP

S3: employers fire employees .

T3:    NNS    VB    NNS    STOP

# Transition diagram



Tag Translation Matrix

	SECOND TAG							
	*	DT	NNS	VB	NN	IN	STOP	
FIRST TAG	*	2/3	1/3					
	DT		2/4	1/4	1/4			
	NNS				3/4			1/4
	VB	1/4	1/4			1/4	1/4	
	NN							1
	IN	1						
	STOP							

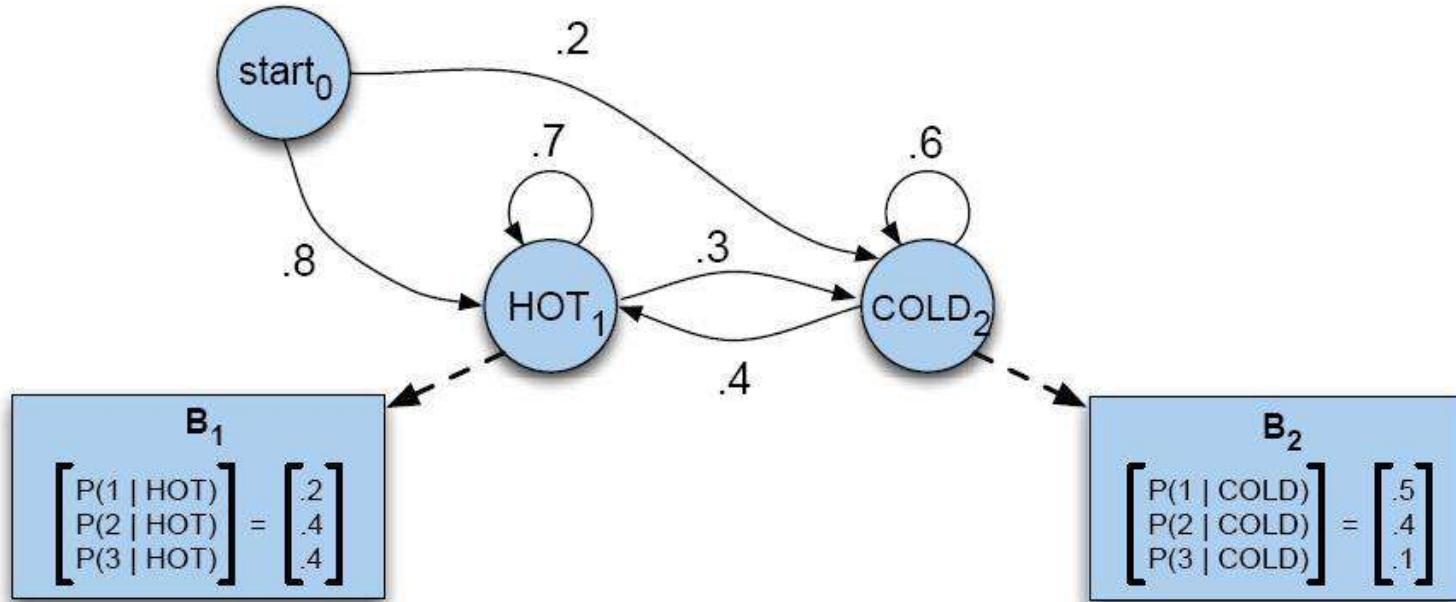
# HMMs for Ice Cream

- You are a climatologist in the year 2799 studying global warming
- You can't find any records of the weather in Baltimore for summer of 2007
- But you find Jason Eisner's diary which lists how many ice-creams Jason ate every day that summer
- Your job: figure out how hot it was each day



# Eisner Task

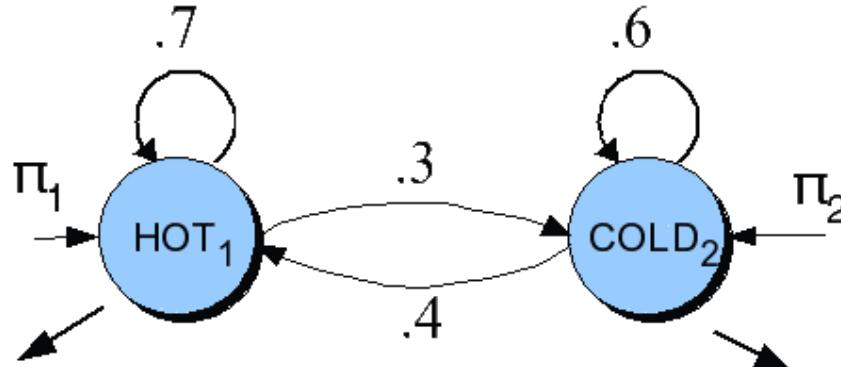
- Given
  - Ice Cream Observation Sequence: 1,2,3,2,2,2,3...
- Produce:
  - Hidden Weather Sequence:  
H,C,H,H,H,C, C...



**What's the state sequence for the observed sequence  
“1 3 1”?**

# HMM for Ice Cream

$$\pi = [.8, .2]$$

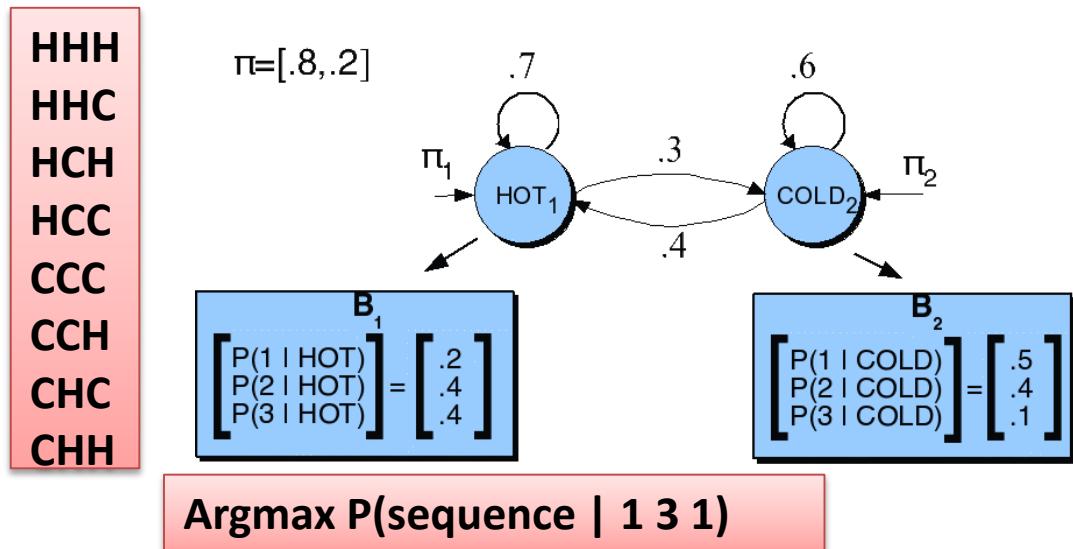


$$B_1 \begin{bmatrix} P(1 | HOT) \\ P(2 | HOT) \\ P(3 | HOT) \end{bmatrix} = \begin{bmatrix} .2 \\ .4 \\ .4 \end{bmatrix}$$

$$B_2 \begin{bmatrix} P(1 | COLD) \\ P(2 | COLD) \\ P(3 | COLD) \end{bmatrix} = \begin{bmatrix} .5 \\ .4 \\ .1 \end{bmatrix}$$

# Ice Cream HMM

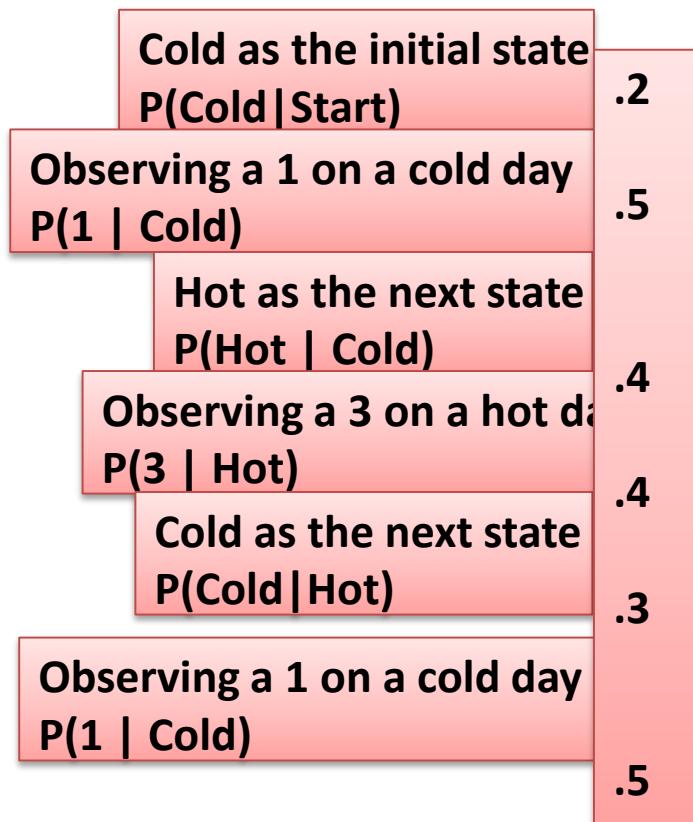
- Let's just do **131** as the sequence
  - How many underlying state (hot/cold) sequences are there?



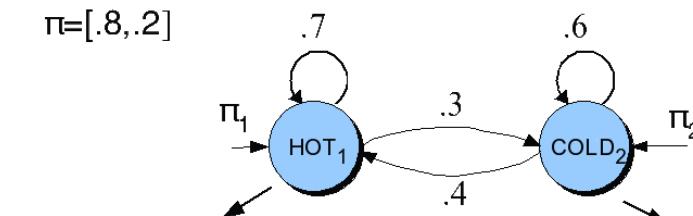
- How do you pick the right one?

# Ice Cream HMM

Let's just do 1 sequence: CHC



$$\pi = [.8, .2]$$

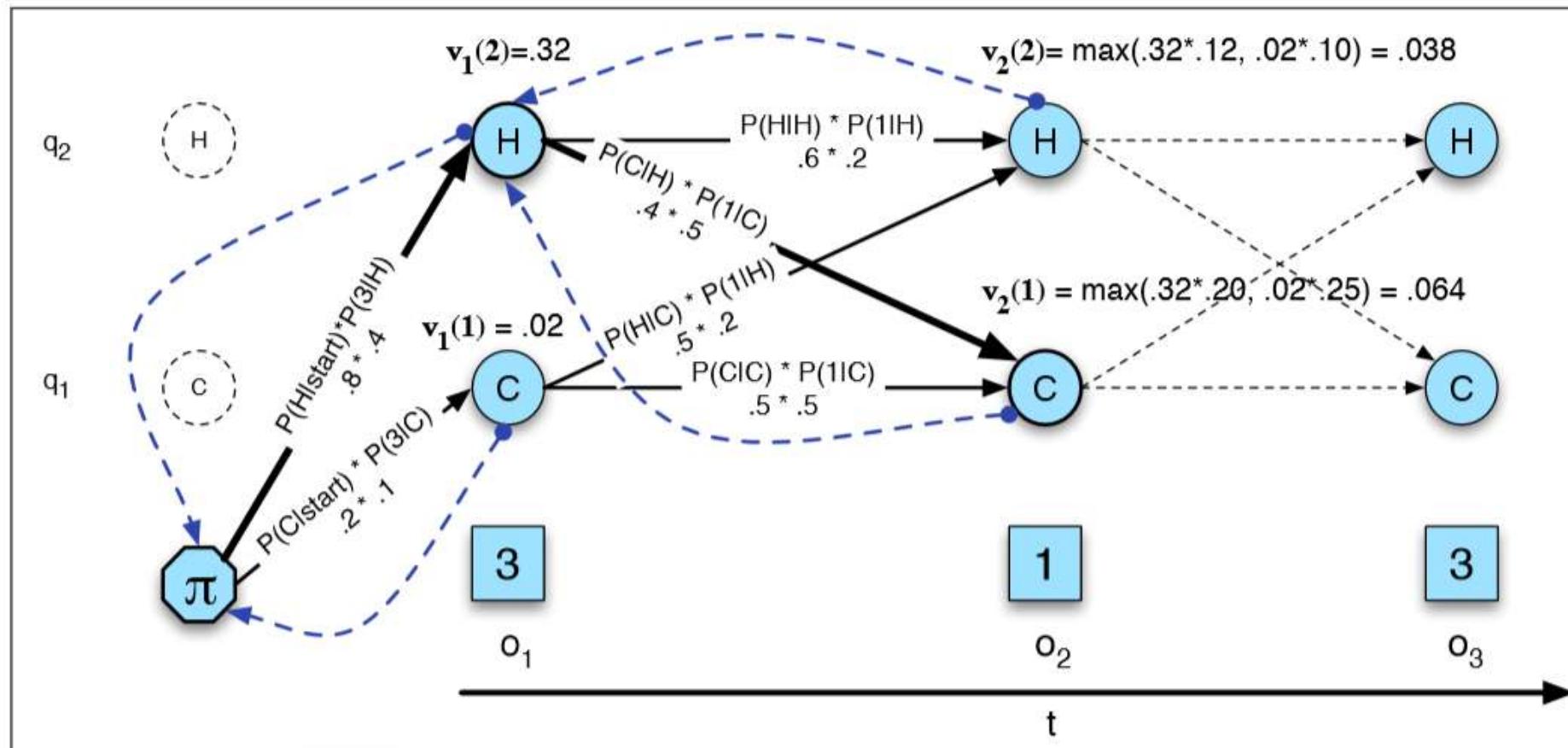


$$B_1 = \begin{bmatrix} P(1 | \text{HOT}) \\ P(2 | \text{HOT}) \\ P(3 | \text{HOT}) \end{bmatrix} = \begin{bmatrix} 0.2 \\ 0.4 \\ 0.4 \end{bmatrix}$$

$$B_2 = \begin{bmatrix} P(1 | \text{COLD}) \\ P(2 | \text{COLD}) \\ P(3 | \text{COLD}) \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.4 \\ 0.1 \end{bmatrix}$$

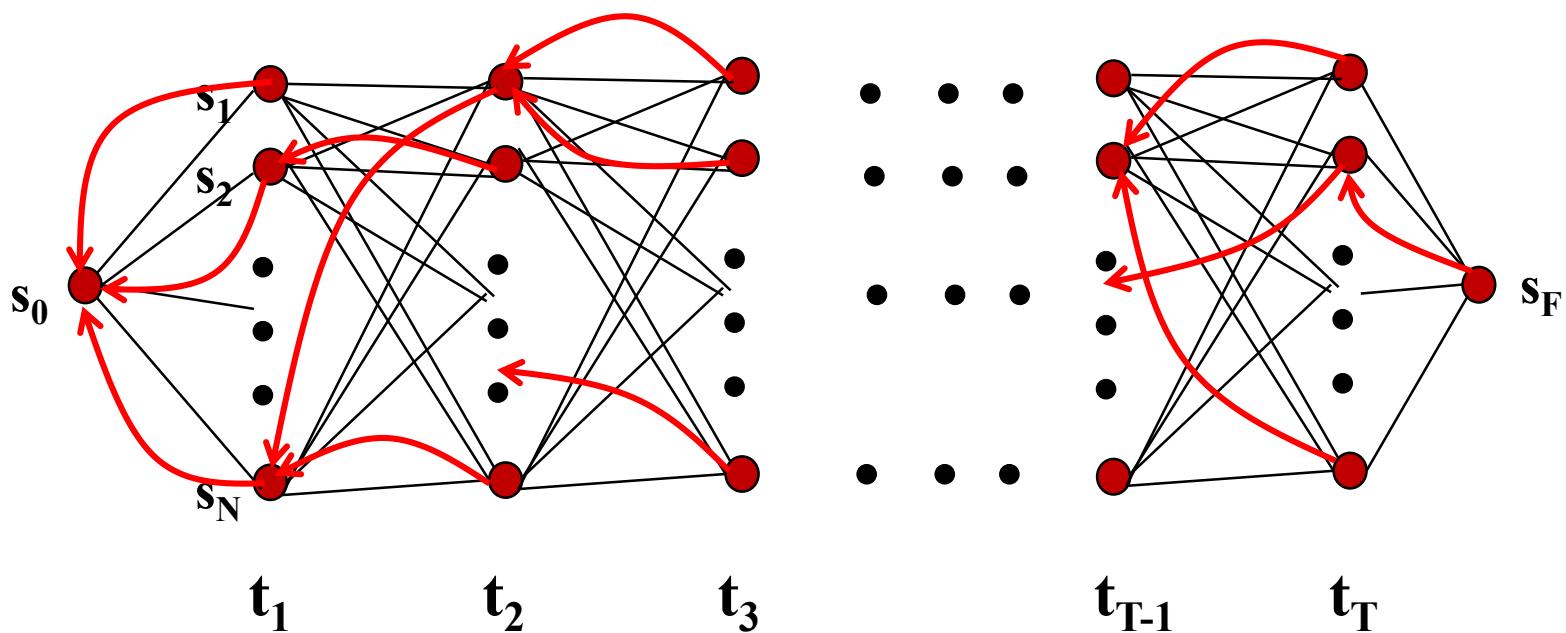
$$\begin{aligned} P(\text{C H C}) &= 0.2 * 0.5 * 0.4 * 0.4 * 0.3 * 0.5 \\ &= 0.0024 \end{aligned}$$

# Viterbi Example 2: Ice Cream

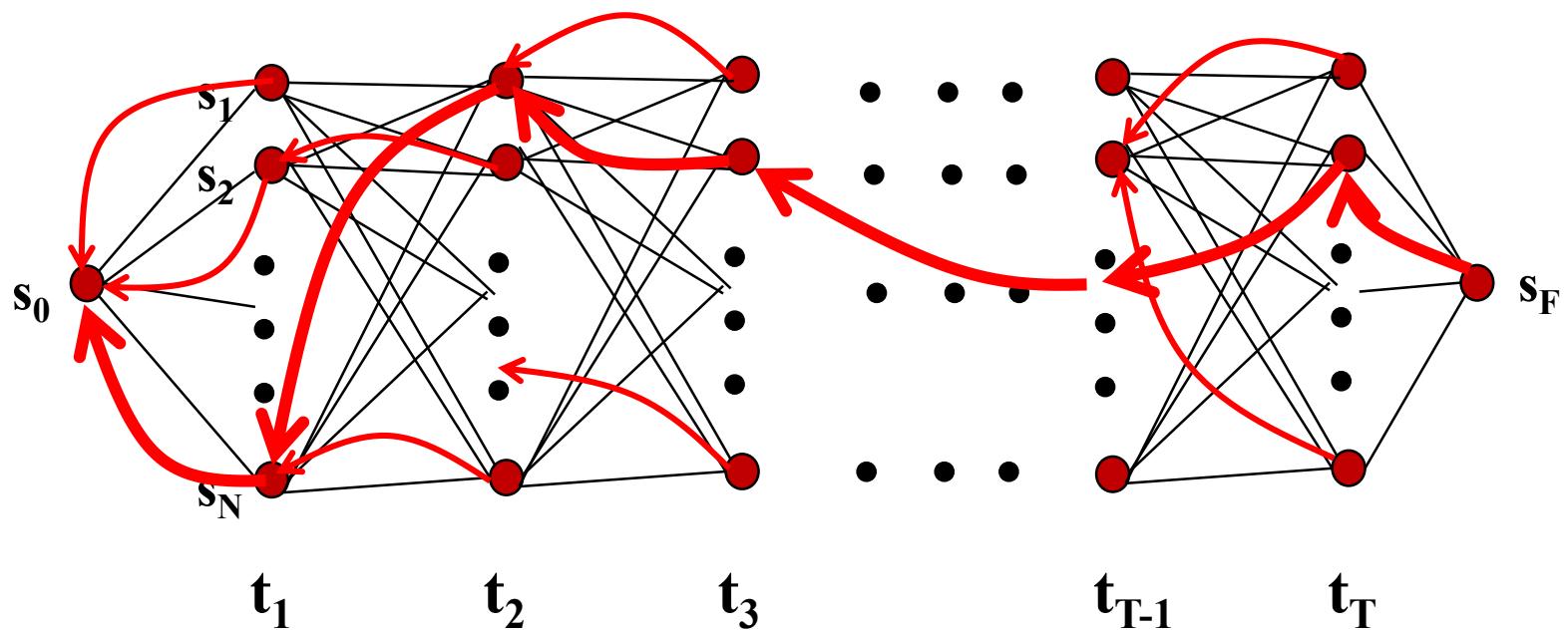


**Figure A.10** The Viterbi backtrace. As we extend each path to a new state account for the next observation, we keep a backpointer (shown with broken lines) to the best path that led us to this state.

# Viterbi Backpointers



# Viterbi Backtrace



Most likely Sequence:  $s_0 \ s_N \ s_1 \ s_2 \dots s_2 \ s_F$

# The Viterbi Algorithm

**function** VITERBI(*observations* of len  $T$ ,*state-graph* of len  $N$ ) **returns** *best-path*

create a path probability matrix  $viterbi[N+2,T]$

**for** each state  $s$  **from** 1 **to**  $N$  **do** ; initialization step

$viterbi[s,1] \leftarrow a_{0,s} * b_s(o_1)$

$backpointer[s,1] \leftarrow 0$

**for** each time step  $t$  **from** 2 **to**  $T$  **do** ; recursion step

**for** each state  $s$  **from** 1 **to**  $N$  **do**

$viterbi[s,t] \leftarrow \max_{s'=1}^N viterbi[s',t-1] * a_{s',s} * b_s(o_t)$

$backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^N viterbi[s',t-1] * a_{s',s}$

$viterbi[q_F,T] \leftarrow \max_{s=1}^N viterbi[s,T] * a_{s,q_F}$  ; termination step

$backpointer[q_F,T] \leftarrow \operatorname{argmax}_{s=1}^N viterbi[s,T] * a_{s,q_F}$  ; termination step

**return** the backtrace path by following backpointers to states back in time from  $backpointer[q_F,T]$



# HMM-POS tagging

- We want, out of all sequences of n tags  $t_1 \dots t_n$  the single tag sequence such that

$$P(t_1 \dots t_n | w_1 \dots w_n) \text{ is highest.}$$

$$\hat{t}_1^n = \operatorname{argmax}_{t_1^n} P(t_1^n | w_1^n)$$

- Hat ^ means “our estimate of the best one”
- Argmax<sub>x</sub> f(x) means “the x such that f(x) is maximized”

# Brute force search over tag sequences

---

- Input sentence <the employees wait for an exam>
- S = { DT NNS VB IN NN }
- All possible tag sequences

DT DT DT DT DT DT STOP

DT DT DT DT DT NNS STOP

DT DT DT DT DT VB STOP

....

# Viterbi algorithm

- Create a table V with N+2 rows and T columns:
  - N – the number of states/tags
  - T – the length of the sequence/sentence
- Initialize the first column
- For each tag t in the tagset compute:

$$V[t, 1] = P(t|start)P(w_1|t)$$

- For each column j = 2 to T in the table V:
  - For each tag t in the tagset compute:

$$V[t, j] = \max_{t'} V[t', j - 1]P(t|t')P(w_j|t)$$

# Example

Transition matrix:  $P(t_i|t_{i-1})$

	NOUN	Verb	Det	Prep	ADV	STOP
<S>	.3	.1	.3	.2	.1	0
Noun	.2	.4	.01	.3	.04	.05
Verb	.3	.05	.3	.2	.1	.05
Det	.9	.01	.01	.01	.07	0
Prep	.4	.05	.4	.1	.05	0
Adv	.1	.5	.1	.1	.1	.1

Emission matrix:  $P(w_i|t_i)$

	a	cat	doctor	in	is	the	very
Noun	0	.5	.4	0	0.1	0	0
Verb	0	0	.1	0	.9	0	0
Det	.3	0	0	0	0	.7	0
Prep	0	0	0	1.0	0	0	0
Adv	0	0	0	.1	0	0	.9

$$V[t, 1] = P(t|start)P(w_1|t)$$

$$V(\text{Noun}, \text{the}) = P(\text{Noun}|<\text{S}>)P(\text{the}|\text{Noun}) = .3 \times 0 = 0$$

$$V(\text{Verb}, \text{the}) = P(\text{Verb}|<\text{S}>)P(\text{the}|\text{Verb}) = .1 \times 0 = 0$$

$$V(\text{Det}, \text{the}) = P(\text{Det}|<\text{S}>)P(\text{the}|\text{Det}) = .3 \times .7 = .21$$

$$V(\text{Prep}, \text{the}) = P(\text{Prep}|<\text{S}>)P(\text{the}|\text{Prep}) = .2 \times .0 = 0$$

$$V(\text{Adv}, \text{the}) = P(\text{Adv}|<\text{S}>)P(\text{the}|\text{Adv}) = .2 \times .0 = 0$$

	w1=the	w2=doctor	w3=is	w4=in	STOP
Noun	0				
Verb	0				
Det	.21				
Prep	0				
Adv	0				

# Example (Contd..)

$$V(\text{Noun}, \text{doctor}) = \max_{t'} V(t', \text{the})XP(\text{Noun}|t')X P(\text{doctor}|\text{Noun}) \\ = \max \{0, 0, .21 (.9 \times .4), 0, 0\} = .0756$$

$$V(\text{Verb}, \text{doctor}) = \max_{t'} V(t', \text{the})XP(\text{Verb}|t')X P(\text{doctor}|\text{Verb}) \\ = \max \{0, 0, .21(.01 \times .1), 0, 0\} = .00021$$

	w1=the	w2=doctor	w3=is	w4=in	STOP
Noun	0	.0756			
Verb	0		.00021		
Det	.21	0			
Prep	0	0			
Adv	0	0			

# Backtracking the Viterbi Matrix

	w1=the	w2=doctor	w3=is	w4=in	STOP
Noun	0	.0756	.001512	0	
Verb	0	.00021	.027216	0	
Det	.21	0	0	0	
Prep	0	0	0	.005443	
Adv	0	0	0	.000272	

Det    Noun    Verb    Prep

# Example: I want to race.

## Viterbi Example

	<b>VB</b>	<b>TO</b>	<b>NN</b>	<b>PPSS</b>
<b>&lt;S&gt;</b>	.019	.0043	.041	.067
<b>VB</b>	.0038	.035	.047	.0070
<b>TO</b>	.83	0	.00047	0
<b>NN</b>	.0040	.016	.087	.0045
<b>PPSS</b>	.23	.00079	.0012	.00014

**Figure 5.15** Tag transition probabilities (the  $\alpha$  array,  $p(t_i|t_{i-1})$ ) computed from the 87-tag Brown corpus without smoothing. The rows are labeled with the conditioning event; thus  $P(PPSS|VB)$  is .0070. The symbol **<s>** is the start-of-sentence symbol.

	<b>I</b>	<b>want</b>	<b>to</b>	<b>race</b>
<b>VB</b>	0	.0093	0	.00012
<b>TO</b>	0	0	.99	0
<b>NN</b>	0	.000054	0	.00057
<b>PPSS</b>	.37	0	0	0

**Figure 5.16** Observation likelihoods (the  $b$  array) computed from the 87-tag Brown corpus without smoothing.

# Viterbi Algorithm Example

---

Algorithm first creates N or four state columns.

- First column corresponds to the observation of the first word “I”,
- the second to the second word “want”,
- the third to the third word “to”, and
- the fourth to the fourth word “race”

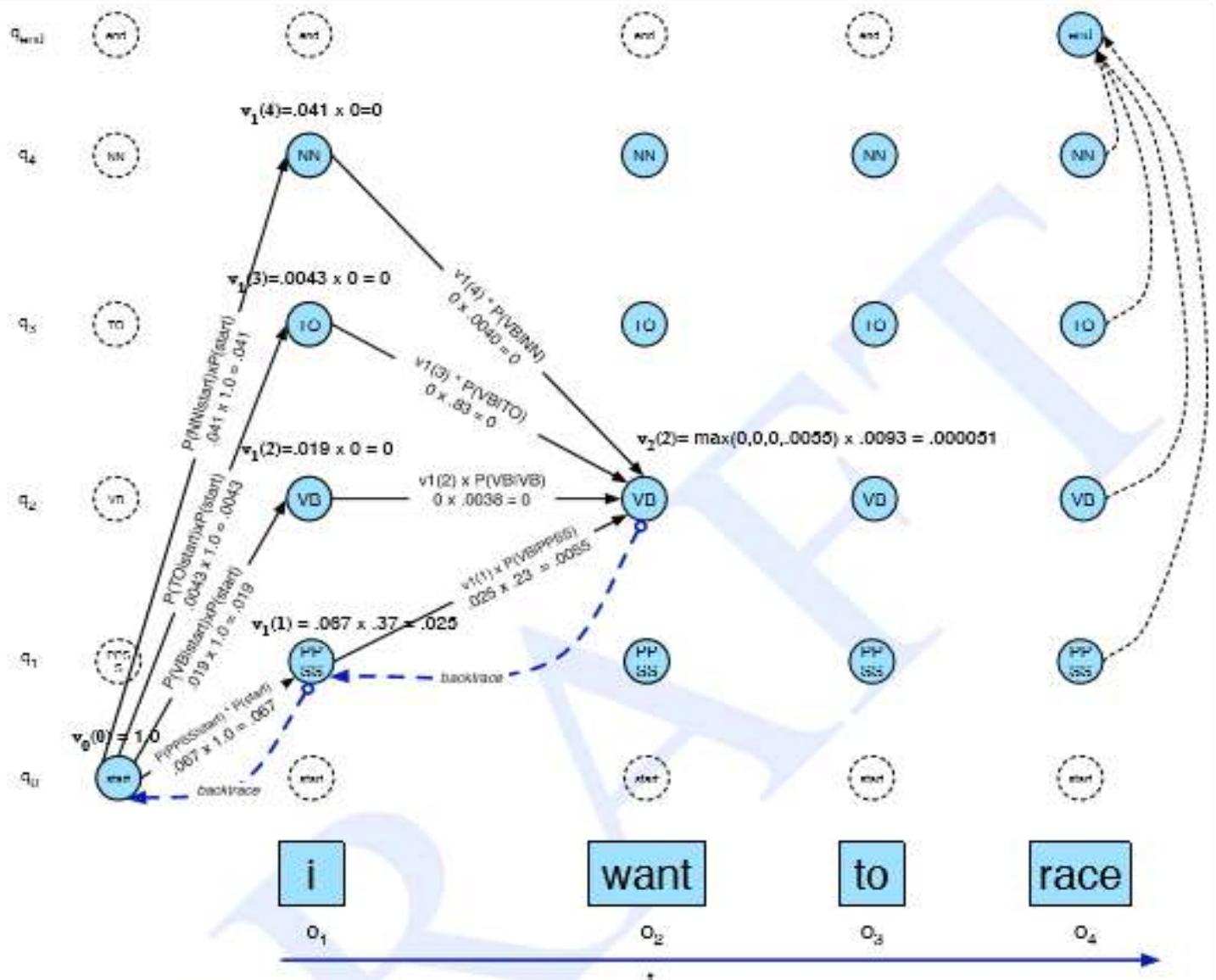
Begin by setting the Viterbi value in each cell to the product of the transition probability (into it from the state state) and the observation probability (of the first word)

# Viterbi Algorithm Example

- For each state  $q_j$  at time  $t$ , the value  $viterbi[s,t]$  is computed by taking the maximum over the extensions of all the paths that lead to the current cell, following the following equation:

$$v_t(j) = \max_{i=1}^N v_{t-1}(i) a_{ij} b_j(o_t)$$

- $v_{t-1}(i)$  the **previous Viterbi path probability** from the previous time step  
 $a_{ij}$  the **transition probability** from previous state  $q_i$  to current state  $q_j$   
 $b_j(o_t)$  the **state observation likelihood** of the observation symbol  $o_t$  given the current state  $j$



**Figure 5.18** The entries in the individual state columns for the Viterbi algorithm. Each cell keeps the probability of the best path so far and a pointer to the previous cell along that path. We have only filled out columns 0 and 1 and one cell of column 2; the rest is left as an exercise for the reader. After the cells are filled in, backtracing from the *end* state, we should be able to reconstruct the correct state sequence PPSS VB TO VB.

# Extending the HMM algorithm to trigrams



$$P(t_1^n) \approx \prod_{i=1}^n P(t_i|t_{i-1}, t_{i-2})$$

$$P(t_i|t_{i-1}, t_{i-2}) = \frac{C(t_{i-2}, t_{i-1}, t_i)}{C(t_{i-2}, t_{i-1})} :$$

- Many of these counts will be zero in any training set
- We will incorrectly predict that a given tag sequence will never occur!
- We need way to estimate  $P(t_i|t_{i-1}, t_{i-2})$  even if the sequence  $t_{i-2}, t_{i-1}, t_i$  never occurs in the training data.
- Solve this problem is to estimate the probability by combining more robust, but weaker estimators

# Extending the HMM algorithm to trigrams



$$\text{Trigrams } \hat{P}(t_i|t_{i-1}, t_{i-2}) = \frac{C(t_{i-2}, t_{i-1}, t_i)}{C(t_{i-2}, t_{i-1})}$$

$$\text{Bigrams } \hat{P}(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

$$\text{Unigrams } \hat{P}(t_i) = \frac{C(t_i)}{N}$$

- Using linear interpolation,
- Estimate the probability  $P(t_i|t_{i-1}t_{i-2})$  by a weighted sum of the unigram, bigram, and trigram probabilities

$$P(t_i|t_{i-1}t_{i-2}) = \lambda_1 \hat{P}(t_i|t_{i-1}t_{i-2}) + \lambda_2 \hat{P}(t_i|t_{i-1}) + \lambda_3 \hat{P}(t_i)$$

Where  $\lambda_1 + \lambda_2 + \lambda_3 = 1$



# C5: Text Mining



**BITS Pilani**  
Hyderabad Campus

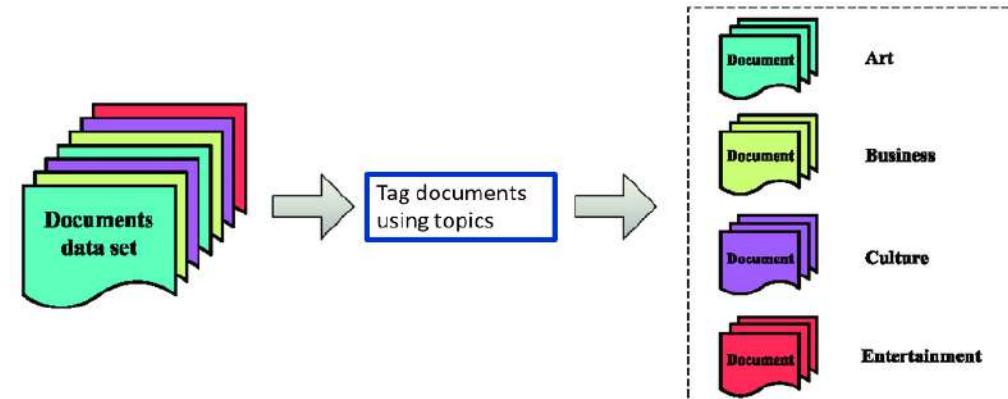
Dr. Chetana Gavankar, Ph.D,  
IIT Bombay-Monash University Australia  
[Chetana.gavankar@pilani.bits-pilani.ac.in](mailto:Chetana.gavankar@pilani.bits-pilani.ac.in)

# Session Content

- 
- Objective of Latent Dirichlet Allocation (LDA)
  - Intuition behind LDA
  - LDA Generative model
  - Mathematical foundations for LDA : Bernoulli Trial, Binomial distribution, Multinomial distribution
  - Mathematical foundations for LDA : Beta distribution, Conjugate Prior and Dirichlet distributions
  - Dirichlet Visualization using Simplex
  - Probabilistic Graphical LDA Model
  - Mathematical modelling of LDA
  - Gibbs Sampling Algorithm
  - Case Study
  - Implementing LDA in Python

# Objectives of Topic Modelling

- Use these annotations to organize, summarize and search the documents.
- Topic Model can be defined as an unsupervised technique to discover topics across various text documents



# Sample output from the LDA

- Four topics learned from the S&P 500 stock market data
- Goal is to find groups of stocks that tend to move together.

Topic 1	Topic 2	Topic 3	Topic 4
Southwestern Energy Range Resources Cabot Oil & Gas EOG Resources Chesapeake Energy Pioneer Resources Devon Energy Peabody Energy Anadarko Petroleum Massey Energy	Penneys Macys Kohls Nordstrom Target Limited Lowes Home Depot American Express Abercrombie	Capital One BNY Mellon Discover Northern Trust Janus JPMorgan Chase State Street Wells Fargo PPL T. Rowe Price	Simon Property Kimco Realty Equity Residential AvalonBay Communities Apartment Investment Vornado Realty Trust Boston Properties Public Storage Host Hotels HCP Inc.

- The topic model does not provide any label to these group of words.

# Sample output from the LDA

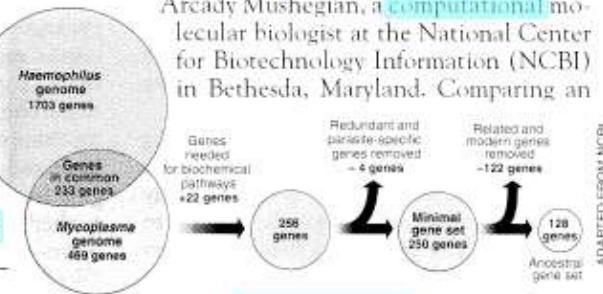
## Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK—How many genes does an organism need to survive? Last week at the genome meeting here,\* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

Although the numbers don't match precisely, those predictions

"are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game, particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains

Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an



**Stripping down.** Computer analysis yields an estimate of the minimum modern and ancient genomes.

\* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

Genetics

Evolutionary biology

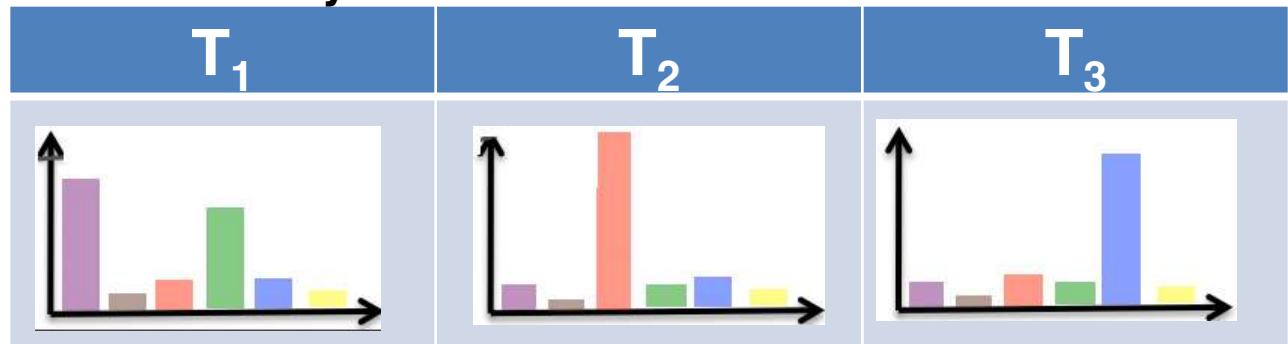
Data Analysis

# Overall schematic

**Corpus**

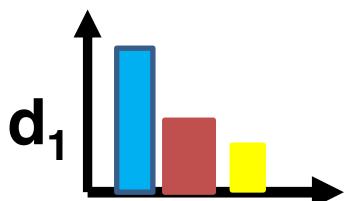


Each topic is defined as a Multinomial distribution over the vocabulary



Documents( $d_1 \dots d_n$ )

K-Topics (Hyper Parameter)

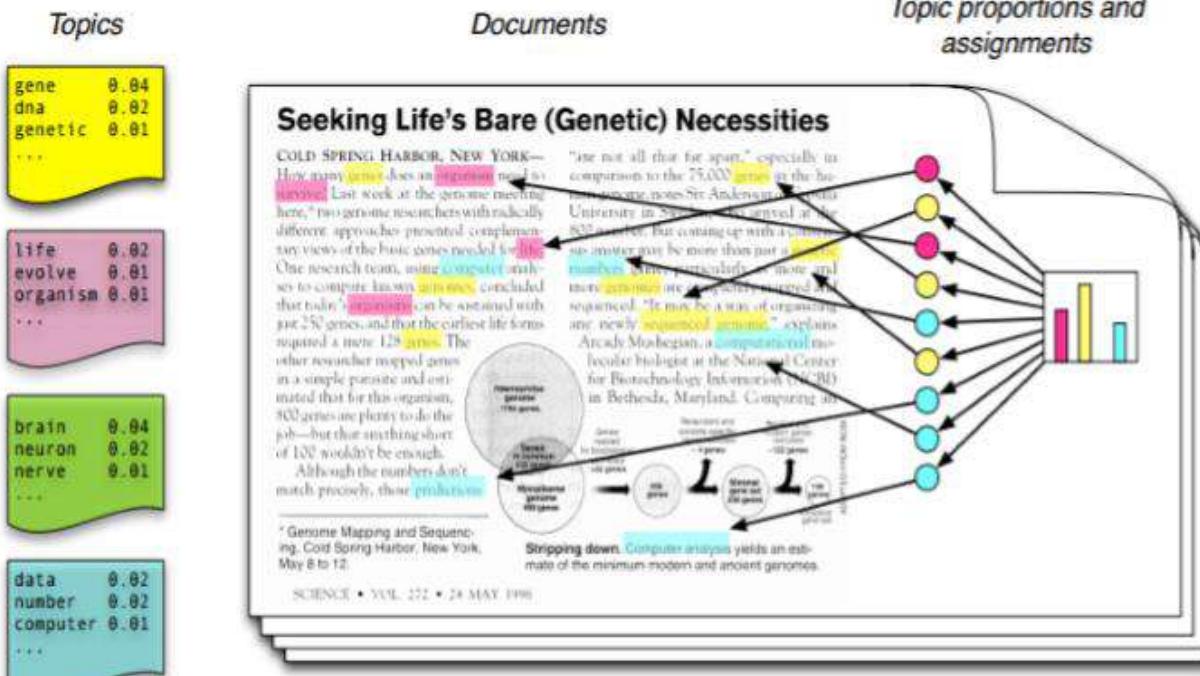


Each document has a distribution over K topics



Vocabulary( $W_1 \dots W_m$ )

# LDA Generative model



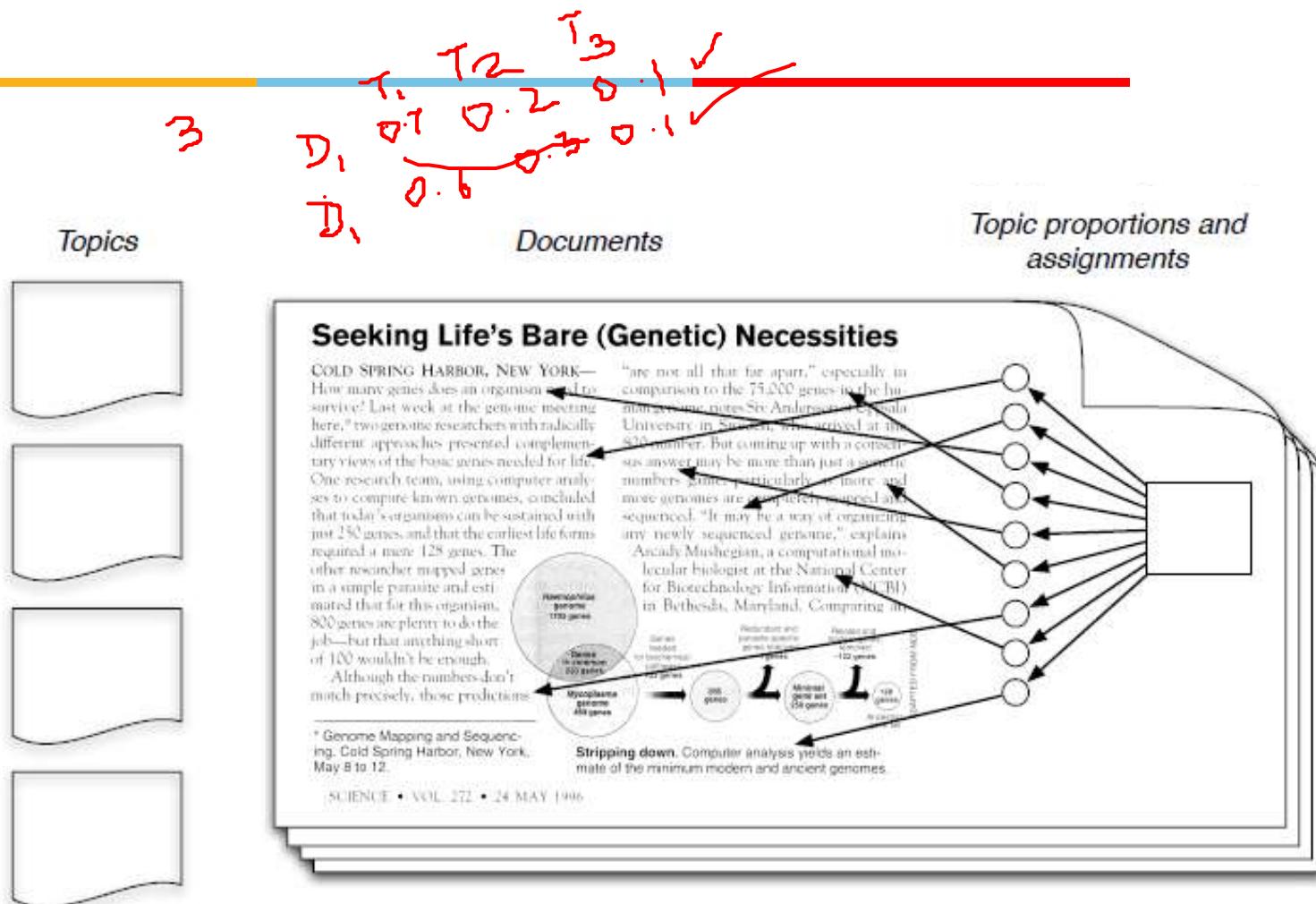
- Each **topic** is a distribution over words
- Each **document** is a mixture of corpus-wide topics
- Each **word** is drawn from one of these topics
- We only observe the words within the documents and the other structure are **hidden variables**.

# The posterior distribution

innovate

achieve

lead

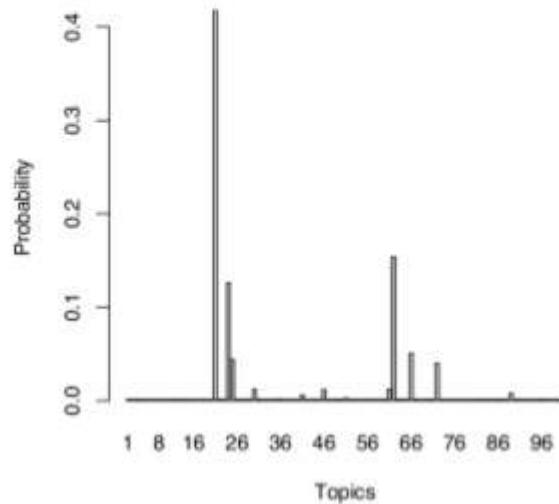


# LDA model

A 100-topic LDA model was fitted to **17,000 articles from the *Science* journal**.

At right are **the top 15 most frequent words** from the most frequent topics.

At left are the **inferred topic proportions** for the example article from previous slide.



“Genetics”	“Evolution”	“Disease”	“Computers”
human	evolution	disease	computer
genome	evolutionary	host	models
dna	species	bacteria	information
genetic	organisms	diseases	data
genes	life	resistance	computers
sequence	origin	bacterial	system
gene	biology	new	network
molecular	groups	strains	systems
sequencing	phylogenetic	control	model
map	living	infectious	parallel
information	diversity	malaria	methods
genetics	group	parasite	networks
mapping	new	parasites	software
project	two	united	new
sequences	common	tuberculosis	simulations

# LDA Generative Process

---

LDA assumes that new documents are created in the following way:

1. Determine number of words in document
2. Choose a topic mixture for the document over a fixed set of topics (i.e. 20% topic A, 30% topic B, 50% topic C)
3. Generate the words in the document by:
  - First pick a topic based on the document's multinomial distribution above.
  - Next pick a word based on the topic's multinomial distribution.

# LDA Generative Process

-Say we have a group of articles and we assume that all of those articles can be characterized by three topics: Animals, Cooking, and Politics.

-Each of those topics can be described by the following words:

- Animals: dog, chicken, cat, nature, zoo
- Cooking: oven, food, restaurant, plates, taste, delicious
- Politics: Republican, Democrat, Congress, ineffective, divisive

-Say we want to generate a new document that is 80% about animals and 20% about cooking.

1. We choose the length of the article (say, 1000 words)
2. We choose a topic based on our specified mixture (so, out of our 1000 words, roughly 800 will come from the topic "animals")
3. We choose a word based on the word distribution for each topic (i.e.

# Intuition behind LDA

---

- Suppose you have a corpus of documents
  - You want LDA to learn the topic representation of K topics in each document and the word distribution of each topic.
  - LDA backtracks from the document level to identify topics that are likely to have generated the corpus.
-

# Intuition behind LDA

- Randomly assign each word in each document to one of the K topics.
- For each document d:
  - Assume that all topic assignments except for the current one are correct.
  - Calculate two proportions:
    1. Proportion of words in document d that are currently assigned to topic t =  $p(\text{topic } t \mid \text{document } d)$
    2. Proportion of assignments to topic t over all documents that come from this word w =  $p(\text{word } w \mid \text{topic } t)$
  - Multiply those two proportions and assign w a new topic based on that probability.  $p(\text{topic } t \mid \text{document } d) * p(\text{word } w \mid \text{topic } t)$
- Eventually we'll reach a steady state where assignments make sense

# Bernoulli Trial

- Any **single trial** with **two possible outcomes** can be modeled as a **Bernoulli trial**: team wins/loses, pitch is a strike/ball, coin comes up heads or tails, etc.
- A Bernoulli trial uses Bernoulli distribution to calculate the probability of either outcome.



**Bernoulli trial**

# Bernoulli: A Special Case of the Binomial Distribution



**Binomial Trial: Chance of getting n heads in a row(n=3)**



**Bernoulli Trial: Chance of getting a heads on a single flip**

# Bernoulli - Distribution Notation

---

- The probability mass function of the Bernoulli distribution is

$$f(x) = P(X=k) = \theta^k (1-\theta)^{1-k}, \quad k=\{0,1\}$$

- The only parameter of the bernoulli distribution is  $\theta$ , which defines the probability of success during a bernoulli trial.

# Binomial distribution

$$k=0 \\ P(X=0) = \theta^0 (1-\theta)^{1-0} = 1-\theta$$

$\overbrace{\square \square DDD}^{n=5}$

$$k=1 \\ P(X=1) = \theta^1 (1-\theta)^{1-1} = \underline{\theta}$$

$$P(X_1=1, X_2=1, X_3=0) = \theta \times \theta (1-\theta) = \theta^2 (1-\theta)$$

$$P\left(\sum_{i=1}^{n=3} X_i = 2\right) = P(1,1,0) + P(1,0,1) + P(0,1,1) \\ = \theta^2 (1-\theta) + \theta^2 (1-\theta) + \theta^2 (1-\theta) \\ = 3\theta^2 (1-\theta)$$

$$P\left(\sum_{i=1}^n X_i = k\right) = \binom{n}{k} \theta^k (1-\theta)^{n-k} \quad \boxed{\binom{n}{k} = \frac{n!}{k!(n-k)!}}$$

# Binomial distribution

If  $p$  is the probability of heads, the probability of getting exactly  $k$  heads in  $n$  independent yes/no trials is given by the binomial distribution  $Bin(n,p)$ :

$$\begin{aligned} P(k \text{ heads}) &= \binom{n}{k} p^k (1-p)^{n-k} \\ &= \frac{n!}{k!(n-k)!} p^k (1-p)^{n-k} \end{aligned}$$

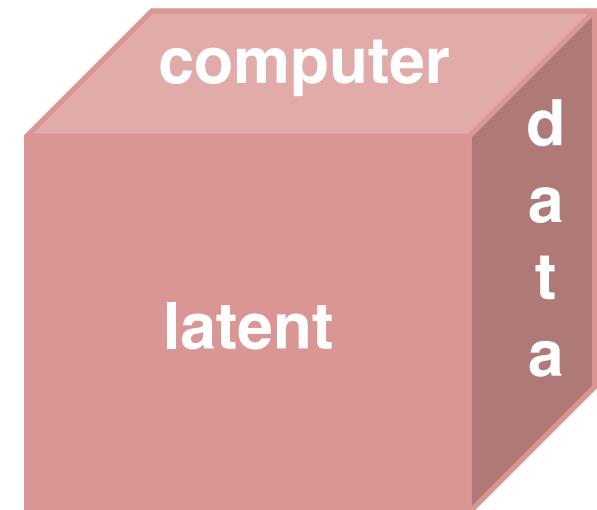
# Multinomial Distribution

- Suppose we roll our die of words having k sides(vocabulary) where each side takes probabilities  $\Theta_1, \dots, \Theta_k$  respectively.
- Probability mass function

$$f(x) = \frac{n!}{x_1!x_2!\cdots x_k!} \theta_1^{x_1} \theta_2^{x_2} \cdots \theta_k^{x_k}$$

k - number of sides on the die

n - number of times the die will be rolled



# Multinomial Distribution

Suppose that we observe an experiment that has  $k$  possible outcomes  $\{O_1, O_2, \dots, O_k\}$  independently  $n$  times.

Let  $p_1, p_2, \dots, p_k$  denote probabilities of  $O_1, O_2, \dots, O_k$  respectively.

Let  $X_i$  denote the number of times that outcome  $O_i$  occurs in the  $n$  repetitions of the experiment.

Then the joint probability function of the random variables  $X_1, X_2, \dots, X_k$  is

$$p(x_1, \dots, x_n) = \frac{n!}{x_1! x_2! \dots x_k!} p_1^{x_1} p_2^{x_2} \dots p_k^{x_k}$$

# Multinomial Distribution

Note:  $p_1^{x_1} p_2^{x_2} \cdots p_k^{x_k}$

is the probability of a sequence of length  $n$  containing

$x_1$  outcomes  $O_1$

$x_2$  outcomes  $O_2$

...

$x_k$  outcomes  $O_k$

$$\frac{n!}{x_1! x_2! \dots x_k!} = \binom{n}{x_1 \quad x_2 \quad \dots \quad x_k}$$

is the number of ways of choosing the positions for the  $x_1$  outcomes  $O_1$ ,  $x_2$  outcomes  $O_2$ , ...,  $x_k$  outcomes  $O_k$

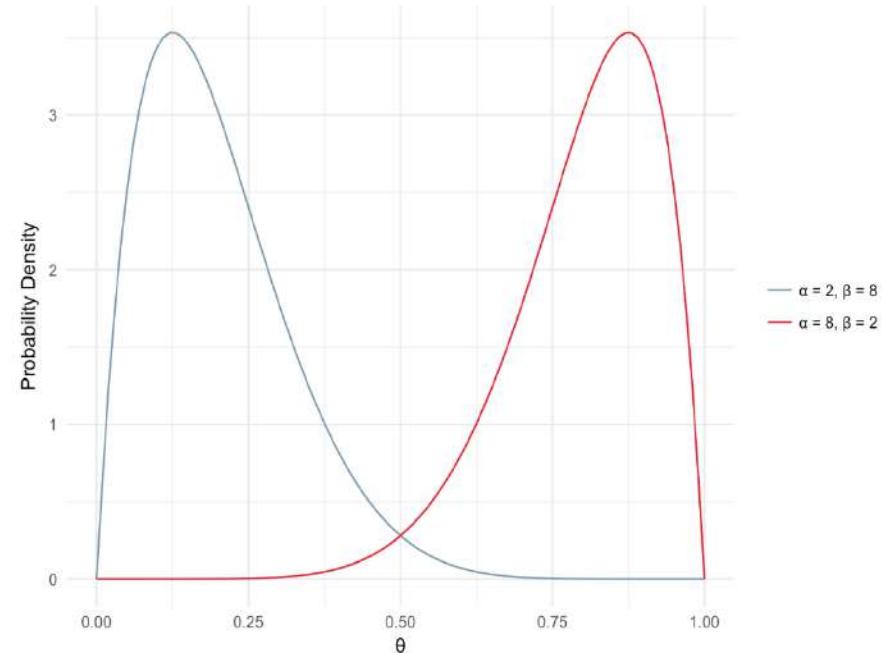
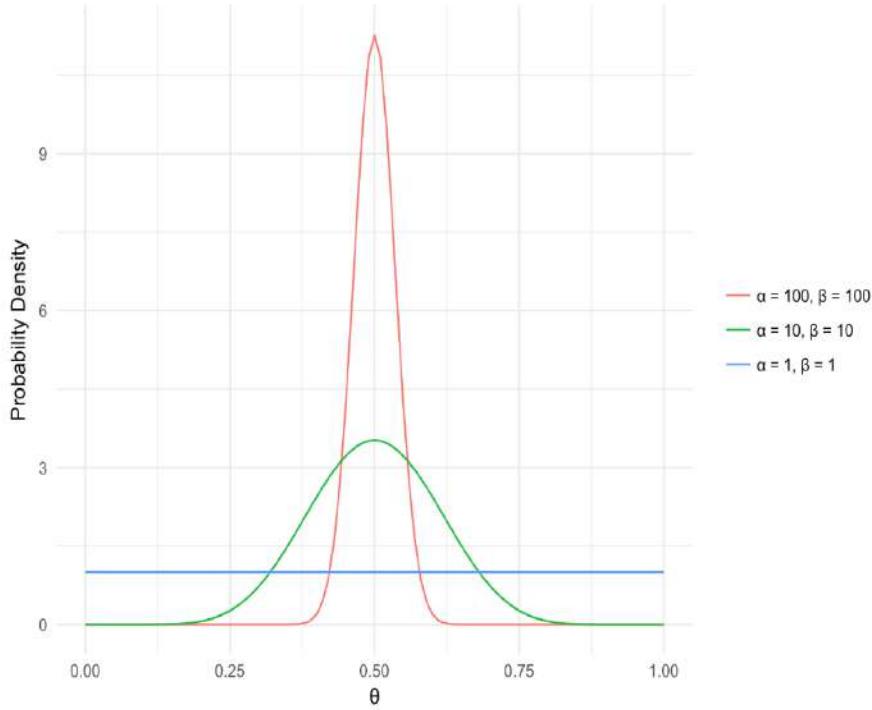
# Beta Distribution

- The probability distribution function for the beta distribution

$$f(\theta; \alpha, \beta) = \frac{\theta^{(\alpha-1)}(1-\theta)^{(\beta-1)}}{B(\alpha, \beta)}$$

# Beta Distribution

- The beta distribution can be thought of as a **probability distribution of probabilities**.



# The Building Blocks of inferring the parameters



- Parameter estimation

$$p(\theta|D) = \frac{\overbrace{p(D|\theta)}^{likelihood} \overbrace{p(\theta)}^{prior}}{\underbrace{p(D)}_{evidence}}$$

# Dirichlet distributions

- Dirichlet distributions are probability distributions over multinomial parameter vectors
  - They are called Beta distributions when k = 2
- The Dirichlet probability density function is defined as

$$Dir(\vec{\theta} | \vec{\alpha}) = \frac{\Gamma(\sum_{i=1}^K \alpha_i)}{\prod_{i=1}^K \Gamma(\alpha_i)} \prod_{i=1}^K \theta_i^{\alpha_i-1}$$

$$\text{➤ } Dir(\vec{\theta} | \vec{\alpha}) = \frac{1}{B(\alpha)} \prod_{i=1}^K \theta_i^{\alpha_i-1} \quad \text{where} \quad \frac{1}{B(\alpha)} = \frac{\Gamma(\sum_{i=1}^K \alpha_i)}{\prod_{i=1}^K \Gamma(\alpha_i)}$$

# Conjugate Prior

In Bayesian probability theory,

- if the posterior distributions  $p(\theta | x)$  are in the same probability distribution family as the prior probability distribution  $p(\theta)$ , the prior and posterior are then called **conjugate distributions**, and the prior is called a **conjugate prior** for the likelihood function

$$p(\theta|D) = \frac{\underbrace{p(D|\theta)}_{\text{likelihood prior}} \underbrace{p(\theta)}_{\text{prior}}}{\underbrace{p(D)}_{\text{evidence}}}$$

- conjugate prior gives a closed-form expression for the posterior; otherwise numerical integration may be necessary.
- Also may give intuition, by more transparently showing how a likelihood function updates a prior distribution

$$\begin{aligned} P(D|\theta) &\sim \text{Normal} \text{ and } P(\theta) \sim \text{Normal} \\ \Rightarrow P(\theta|D) &\sim \text{Normal} \\ P(D|\theta) &\sim \text{Normal} \text{ and } P(\theta) \sim \text{Gamma} \\ \Rightarrow P(\theta|D) &\sim \text{Normal/gamma} \\ P(D|\theta) &\sim \text{Bernoulli} \quad \theta^k (1-\theta)^{N-k} \\ P(\theta) &\sim \theta^{a-1} (1-\theta)^{b-1} \\ \text{Posterior} &\sim \text{Beta} \end{aligned}$$

# Beta distribution as Conjugate Prior for Binomial distribution

Given a **prior**  $P(\theta | \alpha, \beta) = \text{Beta}(\alpha, \beta)$ , and **data**  $D=(H, T)$ , what is our posterior?

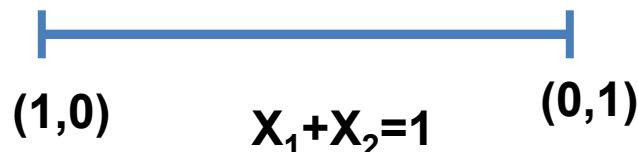
$$\begin{aligned}
 P(\theta | \alpha, \beta, H, T) &\propto P(H, T | \theta)P(\theta | \alpha, \beta) \\
 &\propto \theta^H (1 - \theta)^T \theta^{\alpha-1} (1 - \theta)^{\beta-1} \\
 &= \theta^{H+\alpha-1} (1 - \theta)^{T+\beta-1}
 \end{aligned}$$

With normalization

$$\begin{aligned}
 P(\theta | \alpha, \beta, H, T) &= \frac{\Gamma(H + \alpha + T + \beta)}{\Gamma(H + \alpha)\Gamma(T + \beta)} \theta^{H+\alpha-1} (1 - \theta)^{T+\beta-1} \\
 &= \text{Beta}(\alpha + H, \beta + T)
 \end{aligned}$$

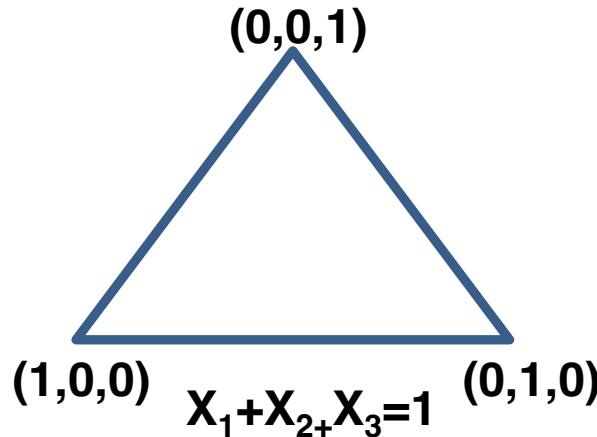
# Visualization of the simplex

- This is often referred as **simplex** and a most convenient way to visualize this is using a certain shapes depending upon the number of topics.
- Suppose K=2 topics which can be modeled as 1-simplex and can be visualized using a line.



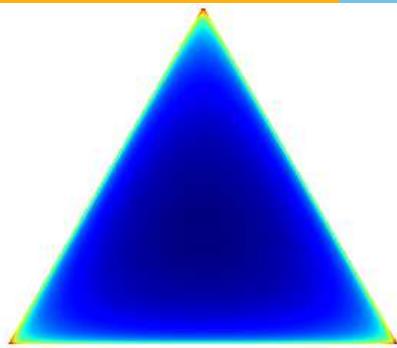
# Visualization of the simplex

- Suppose K=3 topics which can be modeled as 2-simplex and can be visualized using a triangle.

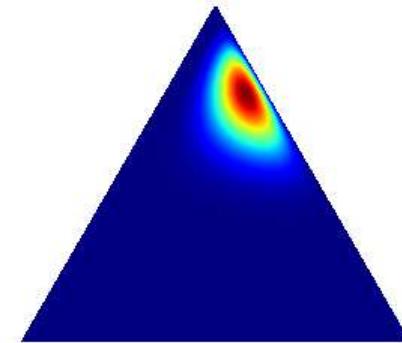


- If we have **K topics** this can be generated using **K-1 simplex**.

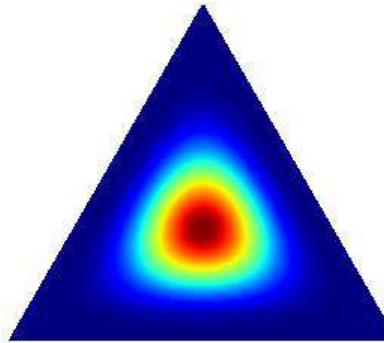
# Shape of the Dirichlet distribution



Dirichlet(0.999, 0.999, 0.999)



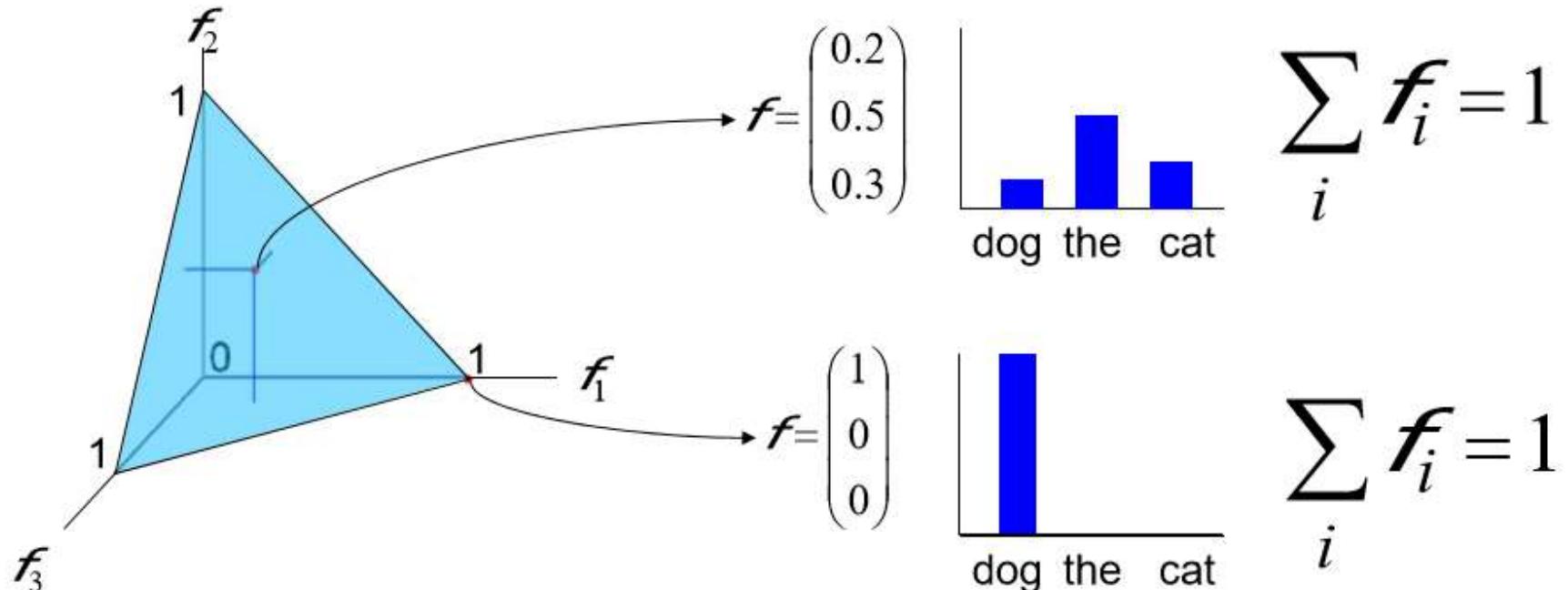
Dirichlet(2, 5, 15)



Dirichlet(5, 5, 5)

# Dirichlet distributions

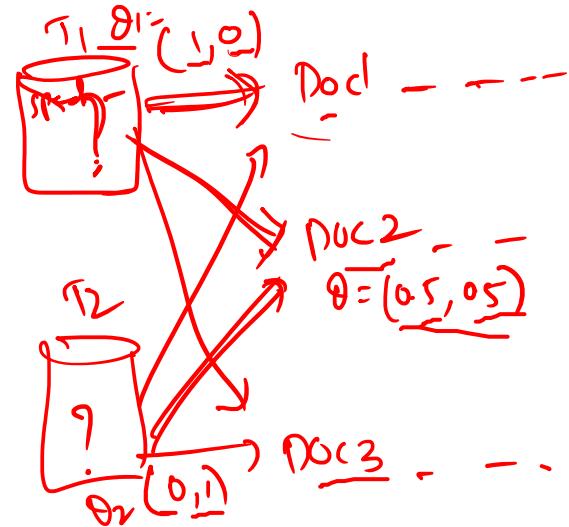
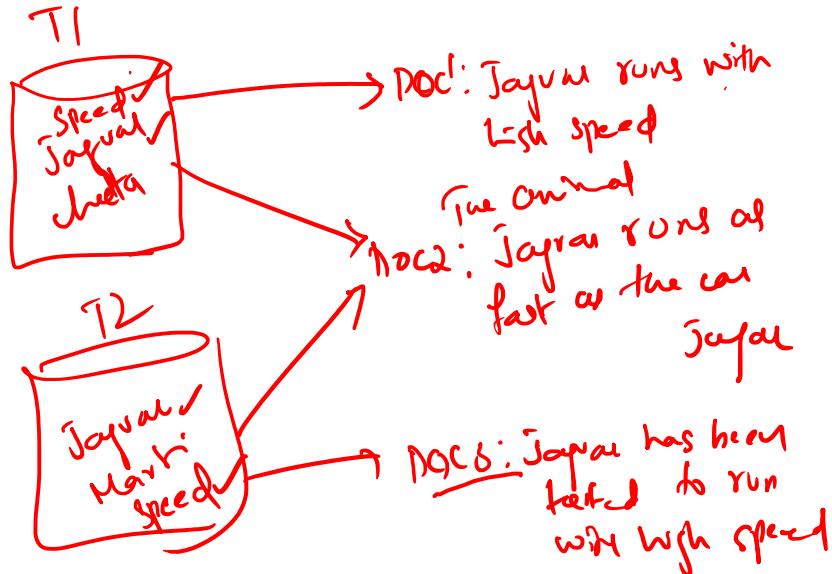
Each point on a  $k$  dimensional simplex is a multinomial probability distribution:



$$\sum_i f_i = 1$$

$$\sum_i f_i = 1$$

# Statistical Inference





---

# Topic Modelling Example - LDA



Sports



Politics



Science



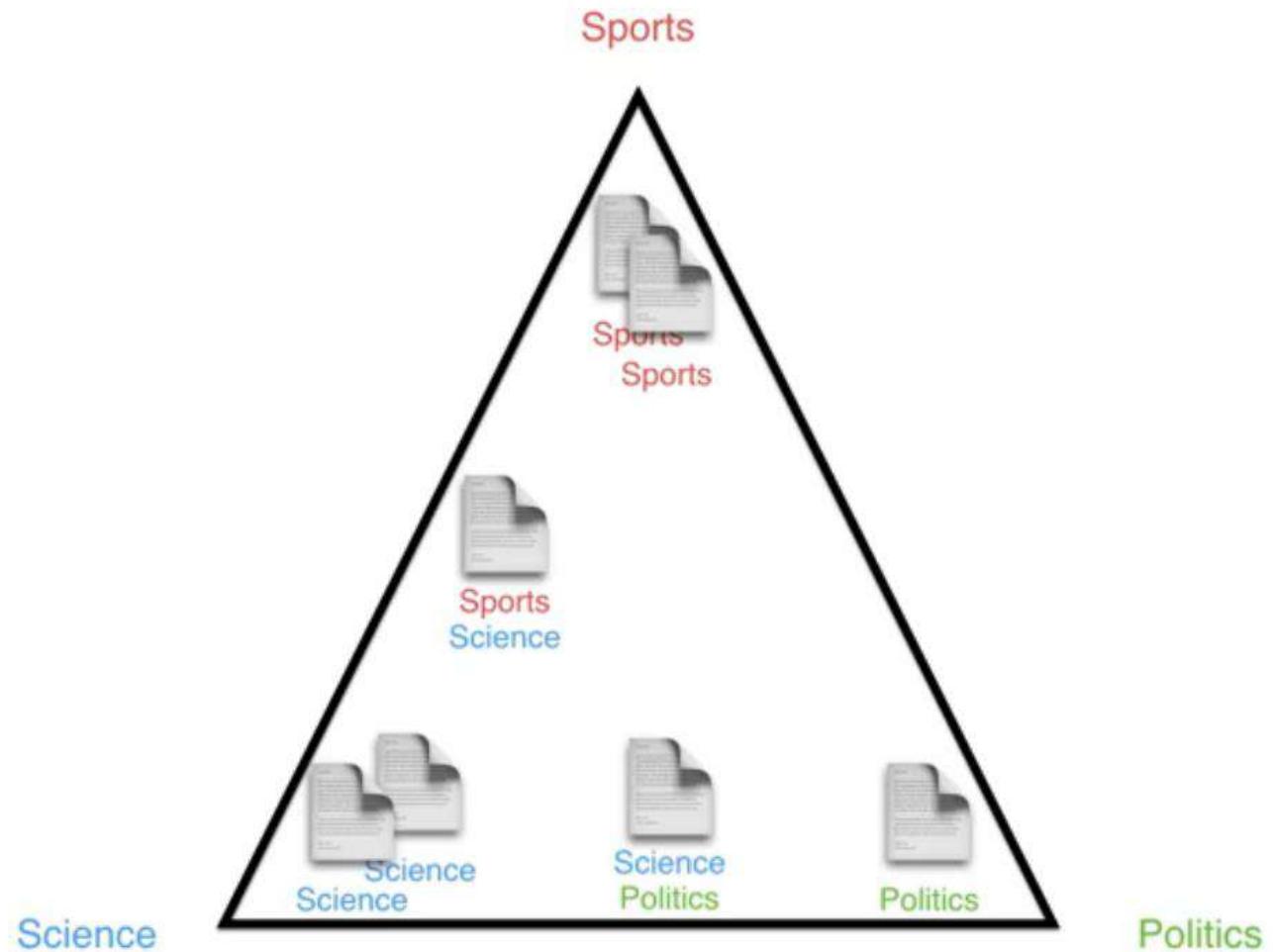
Science

Science

Politics

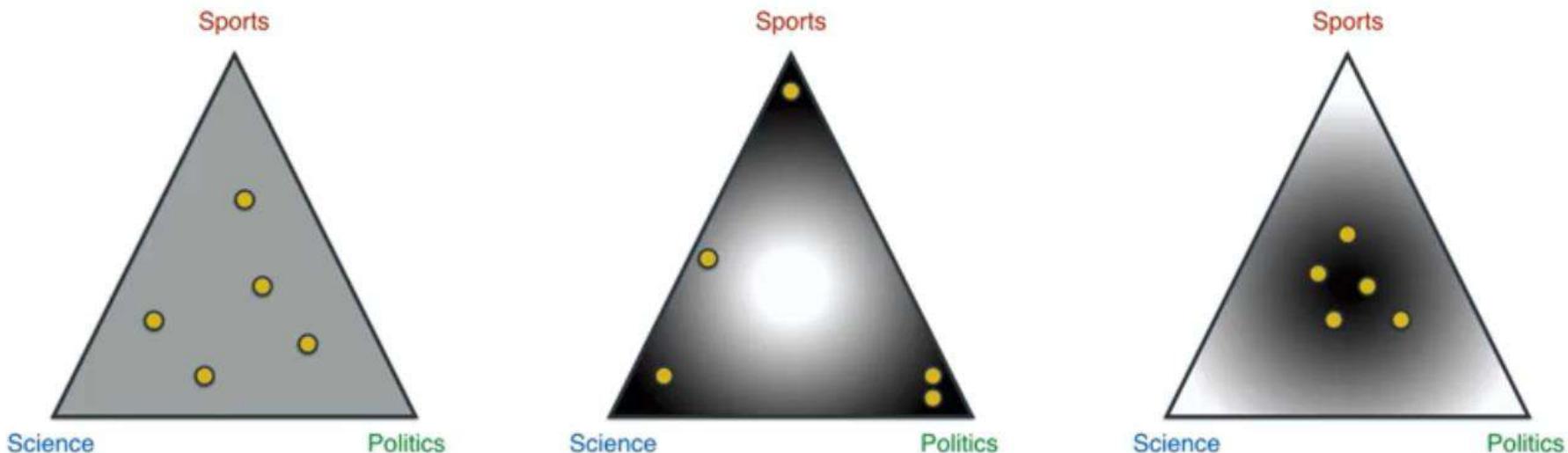
Sports

# LDA

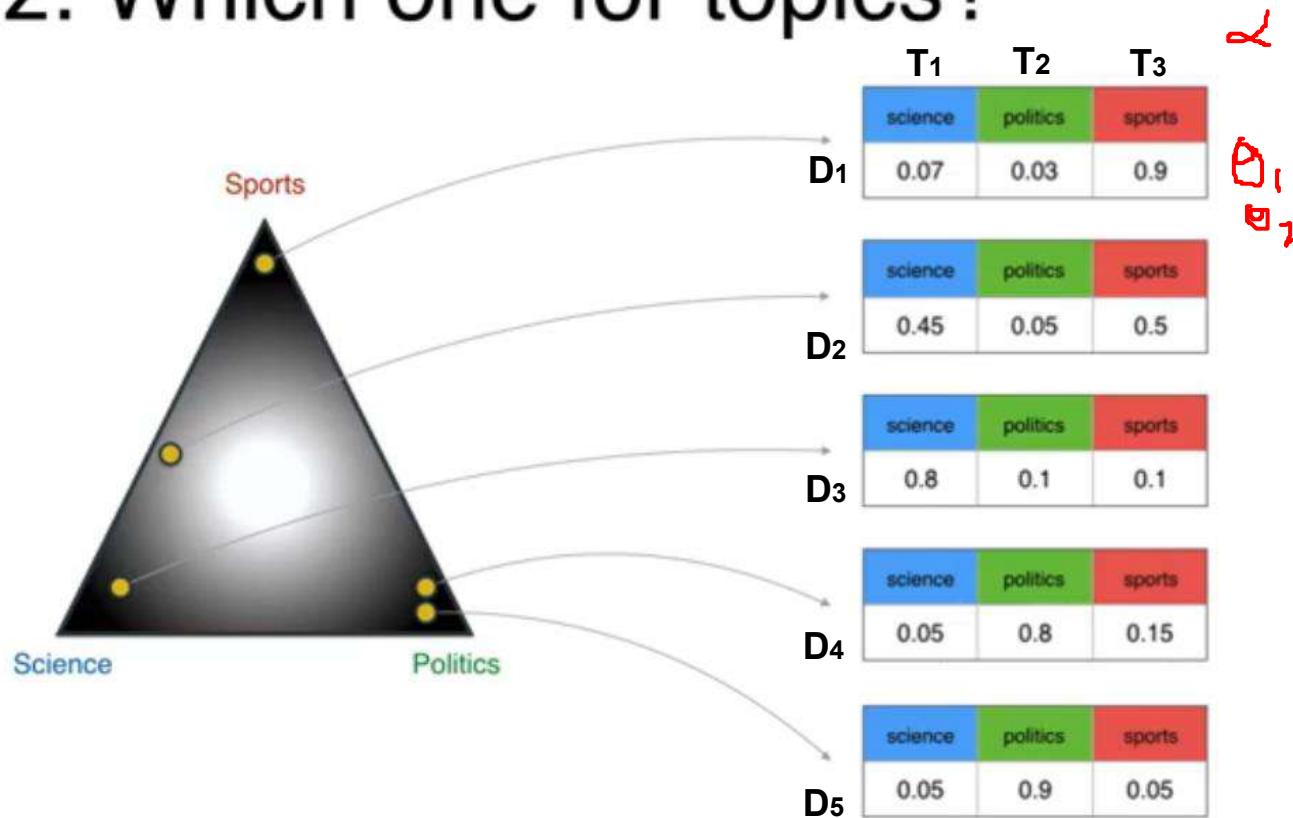


---

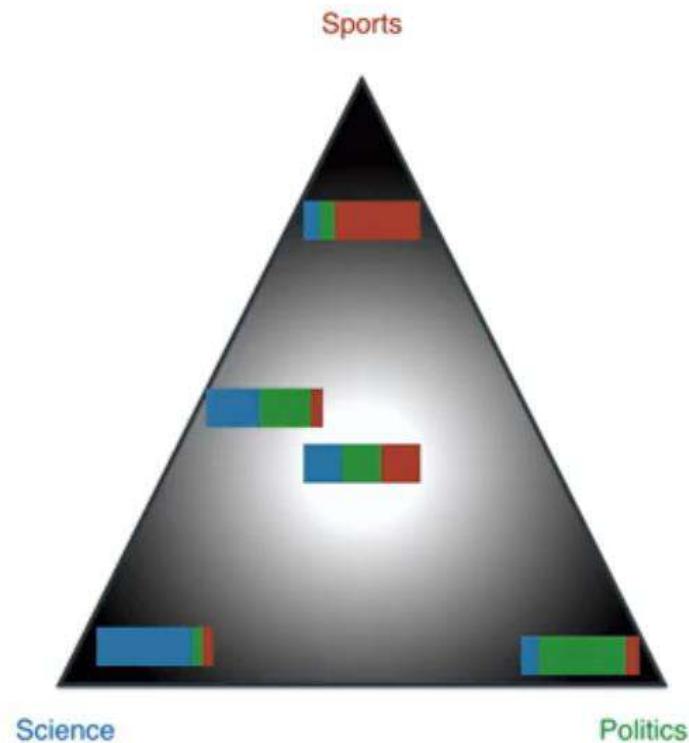
# Quiz: Which one for topics?



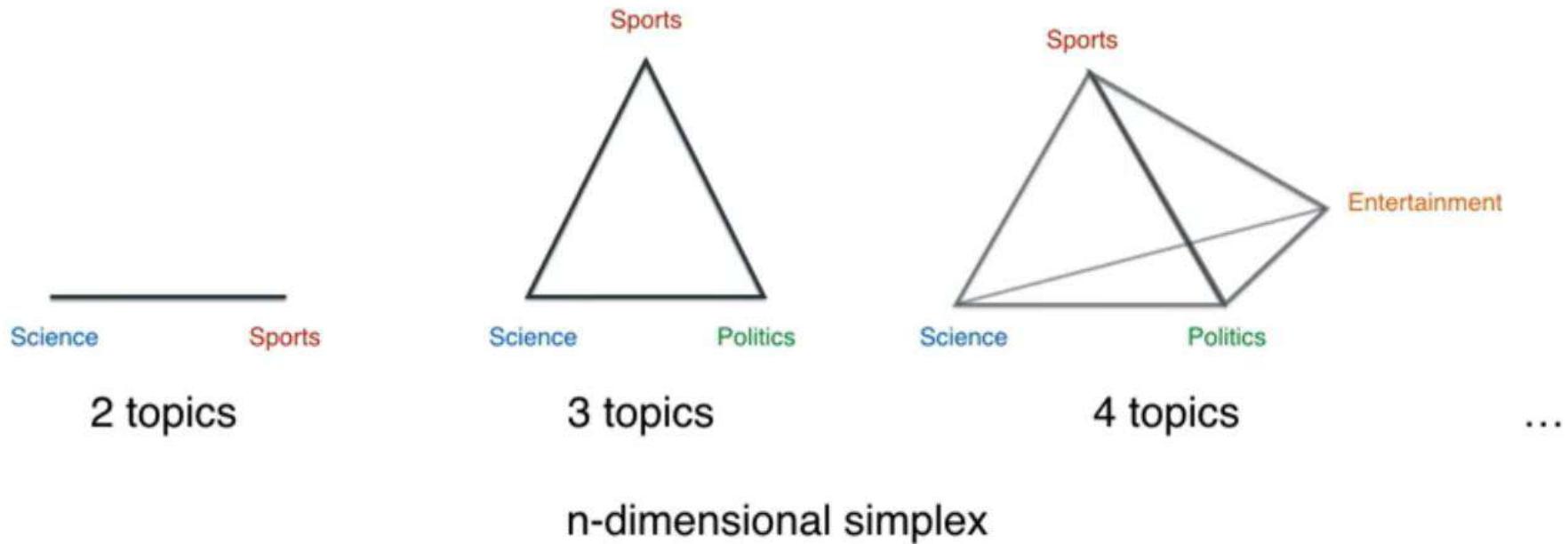
# Quiz: Which one for topics?



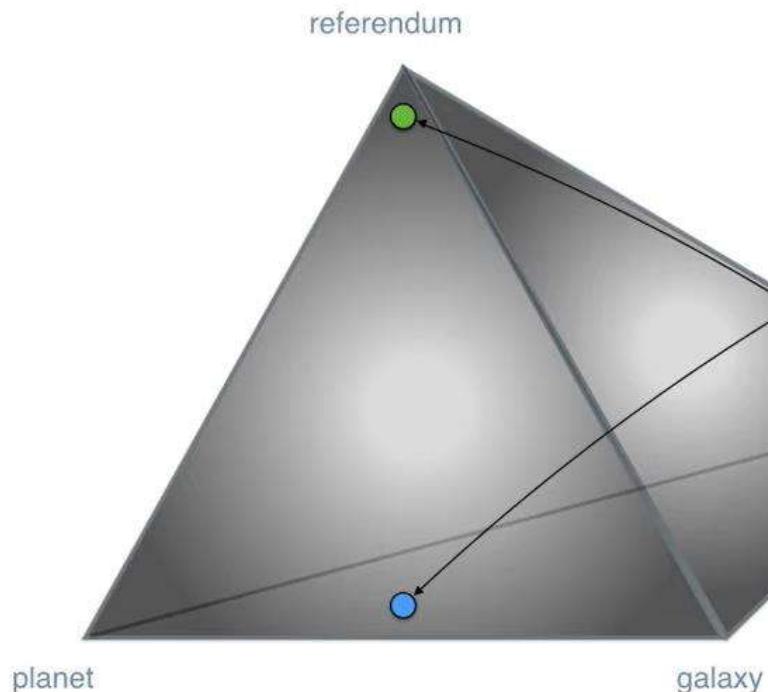
# A distribution of distributions



# More topics? More dimensions



# Word distributions



$T_1$  science

$T_2$  sports

$T_3$  politics

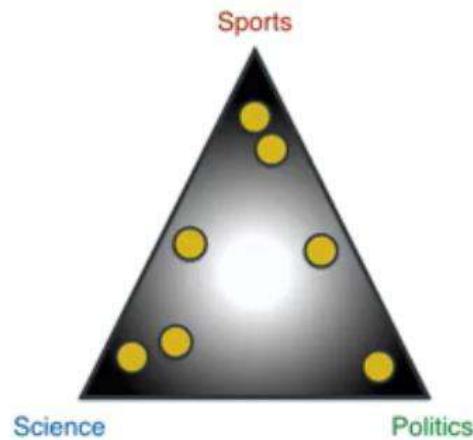
ball

$W_1$	$W_2$	$W_3$	$W_4$
Galaxy	Planet	Ball	Referendum
0.4	0.4	0.1	0.1

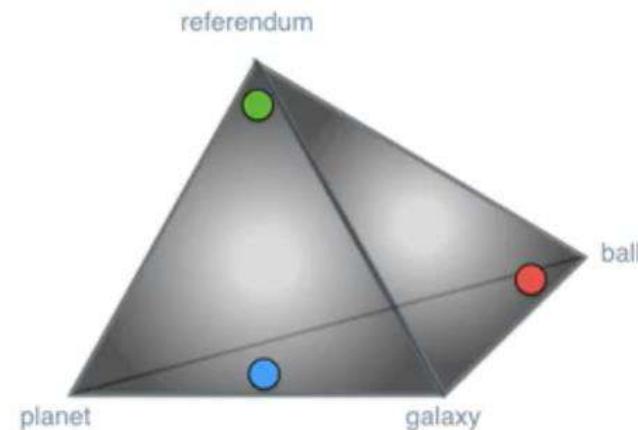
Galaxy	Planet	Ball	Referendum
0.3	0.1	0.5	0.1

Galaxy	Planet	Ball	Referendum
0.1	0.1	0.1	0.7

# Two Dirichlet distributions

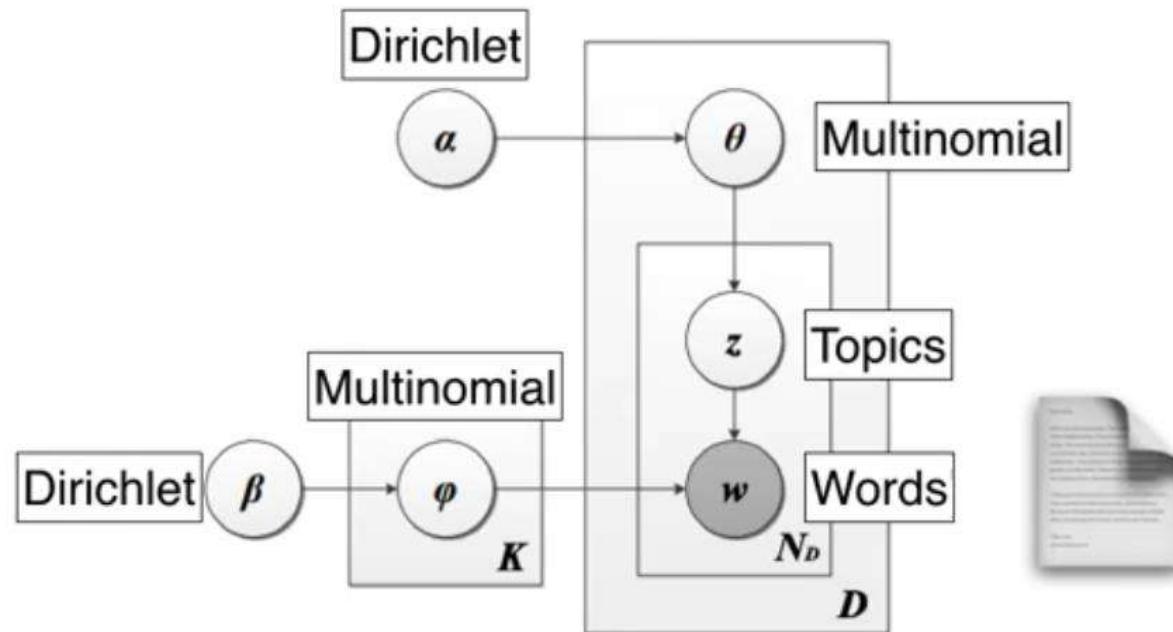


Documents-Topics



Topics-Words

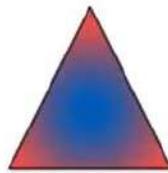
# Blueprint for the LDA machine



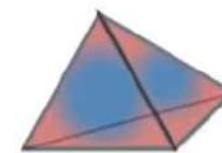
# Mathematical modelling of LDA

## Probability of a document

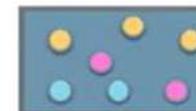
$$P(\mathbf{W}, \mathbf{Z}, \boldsymbol{\theta}, \boldsymbol{\varphi}; \alpha, \beta) = \prod_{j=1}^M P(\theta_j; \alpha) \prod_{i=1}^K P(\varphi_i; \beta) \prod_{t=1}^N P(Z_{j,t} | \theta_j) P(W_{j,t} | \varphi_{Z_{j,t}})$$



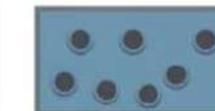
Topics



Words



Topics



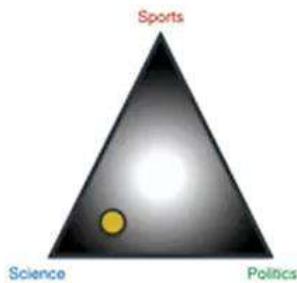
Words

Dirichlet  
Distributions

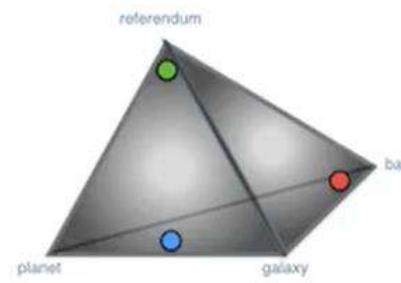
Multinomial  
Distributions

# Mathematical modelling of LDA

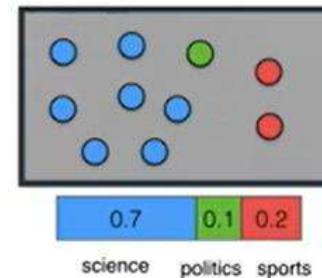
$$\prod_{j=1}^M P(\theta_j; \alpha)$$



$$\prod_{i=1}^K P(\varphi_i; \beta)$$



$$\prod_{t=1}^N P(Z_{j,t} | \theta_j)$$



science	politics	sports
0.7	0.1	0.2

Galaxy	Planet	Ball	Referendum
0.4	0.4	0.1	0.1
Galaxy	Planet	Ball	Referendum
0.1	0.1	0.1	0.7
Galaxy	Planet	Ball	Referendum
0.3	0.1	0.5	0.1

$$P(W_{j,t} | \varphi_{Z_{j,t}})$$

## Topics

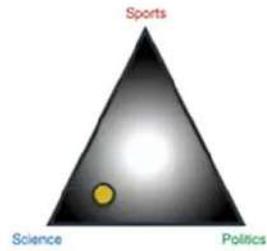
- science → planet
- science → galaxy
- sports → ball
- science → planet
- science → galaxy
- politics → referendum
- sports → galaxy
- sports → ball
- science → referendum

## Words

- planet
- galaxy
- ball
- planet
- galaxy
- referendum
- galaxy
- ball
- referendum

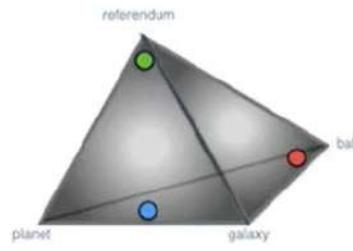
# Mathematical modelling of LDA

$$\prod_{j=1}^M P(\theta_j; \alpha)$$



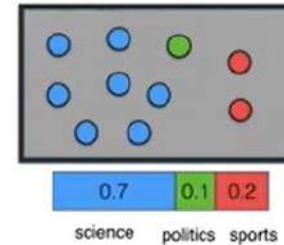
science	politics	sports
0.7	0.1	0.2

$$\prod_{i=1}^K P(\varphi_i; \beta)$$



Galaxy	Planet	Ball	Referendum
0.4	0.4	0.1	0.1
Galaxy	Planet	Ball	Referendum
0.1	0.1	0.1	0.7
Galaxy	Planet	Ball	Referendum
0.3	0.1	0.5	0.1

$$\prod_{t=1}^N P(Z_{j,t} | \theta_j)$$



$$P(W_{j,t} | \varphi_{Z_{j,t}})$$

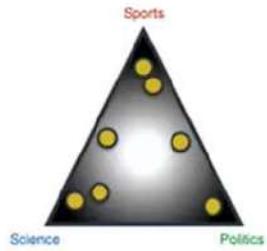
galaxy	galaxy	planet
galaxy	planet	ball
galaxy	planet	planet
referendum	referendum	referendum
planet	ball	referendum
referendum	referendum	referendum
galaxy	referendum	referendum
galaxy	referendum	referendum
planet	ball	galaxy
ball	ball	ball
planet	ball	galaxy
galaxy	galaxy	referendum
referendum	referendum	referendum

Topics  
 science  
 science  
 sports  
 science  
 science  
 politics  
 sports  
 sports  
 science

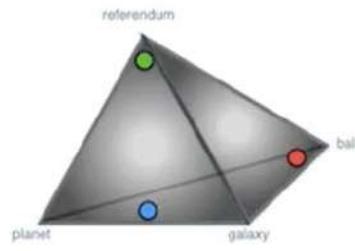
planet
galaxy
ball
planet
galaxy
referendum
galaxy
ball
referendum

# Mathematical modelling of LDA

$$\prod_{j=1}^M P(\theta_j; \alpha)$$



$$\prod_{i=1}^K P(\varphi_i; \beta)$$



$$\prod_{t=1}^N P(Z_{j,t} | \theta_j)$$

$$P(W_{j,t} | \varphi_{Z_{j,t}})$$

$P(\text{same articles}) = \text{low}$

planet  
galaxy  
ball  
planet  
galaxy  
referendum  
galaxy  
ball

ball  
planet  
galaxy  
galaxy  
ball  
ball  
referendum  
ball

referendum  
referendum  
referendum  
referendum  
referendum  
planet  
referendum  
galaxy

referendum  
referendum  
planet  
galaxy  
referendum  
planet  
galaxy  
ball

galaxy  
planet  
galaxy  
ball  
planet  
planet  
galaxy

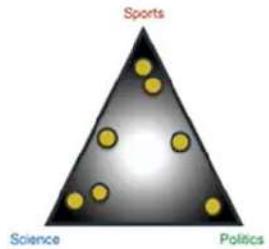
galaxy  
planet  
galaxy  
ball  
referendum  
planet  
ball  
galaxy  
planet

referendum  
galaxy  
ball  
ball  
galaxy  
referendum  
planet  
galaxy

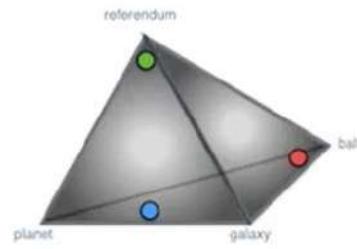


# Mathematical modelling of LDA

$$\prod_{j=1}^M P(\theta_j; \alpha)$$

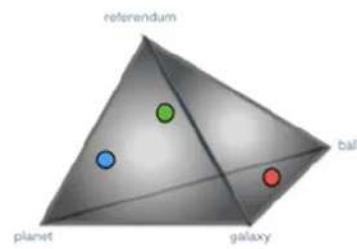
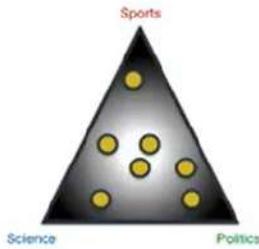


$$\prod_{i=1}^K P(\varphi_i; \beta)$$



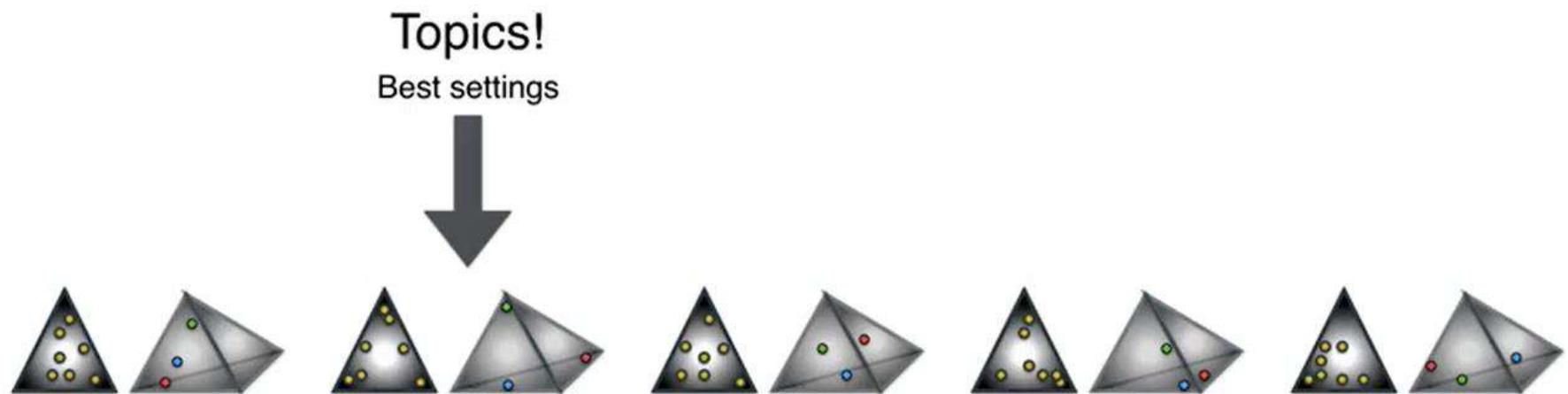
$$\prod_{t=1}^N P(Z_{j,t} | \theta_j) \quad P(W_{j,t} | \varphi_{Z_{j,t}})$$

$P(\text{same articles}) = \text{low}$



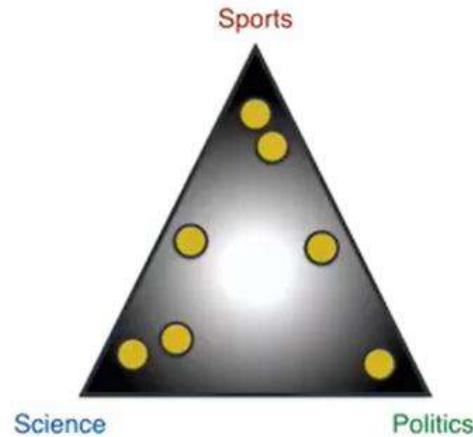
$P(\text{same articles}) = \text{very low}$

# Best settings on the machine

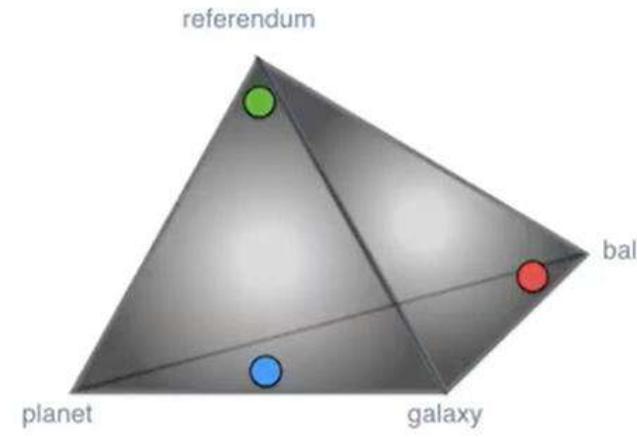


# Mathematical modelling of LDA

## The winning arrangements

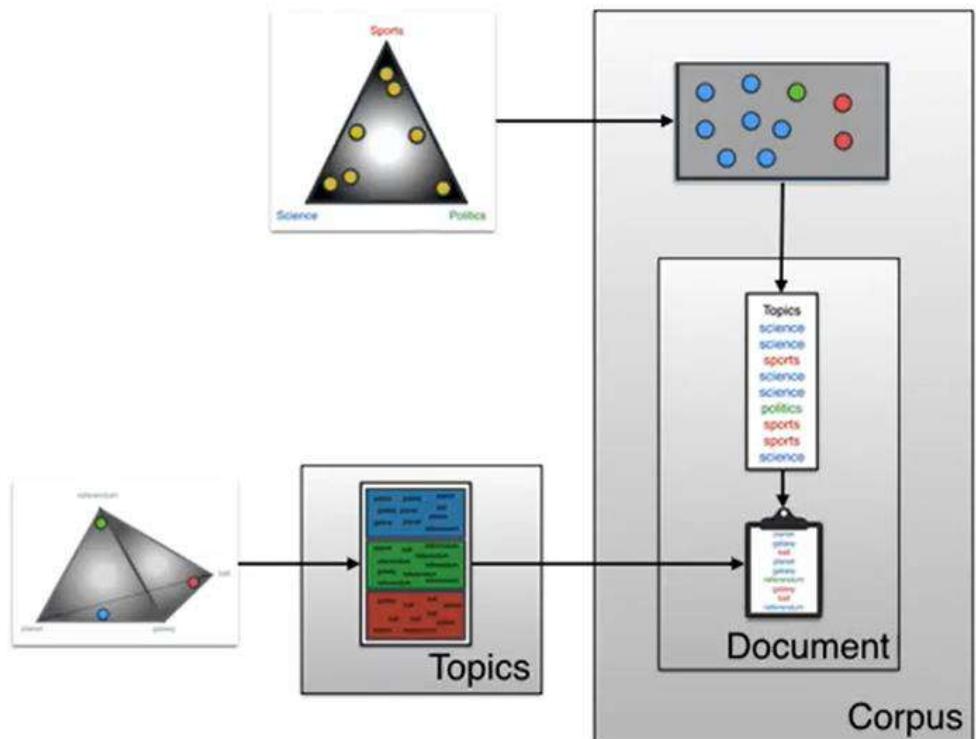
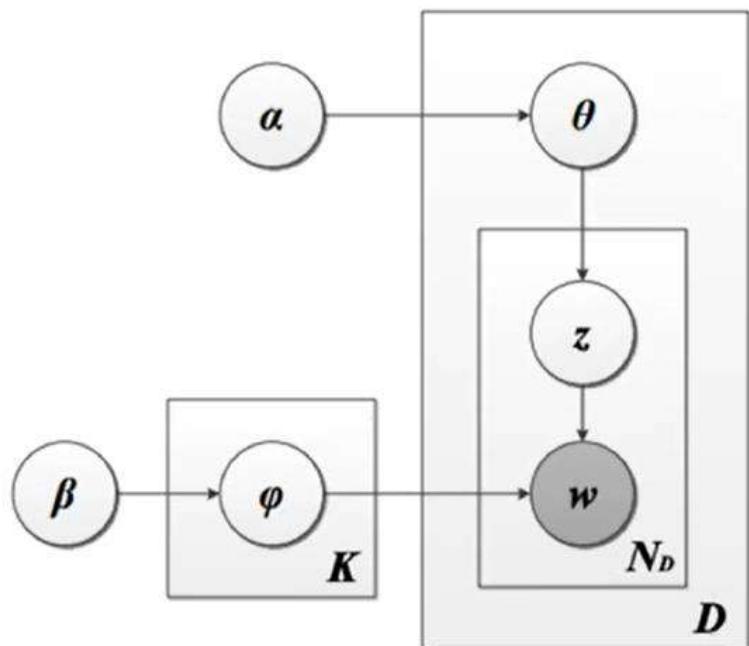


Documents-Topics



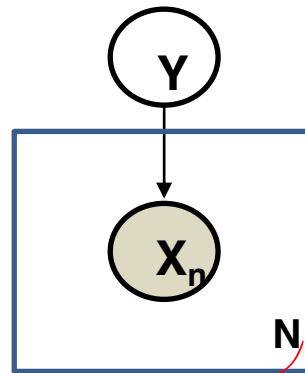
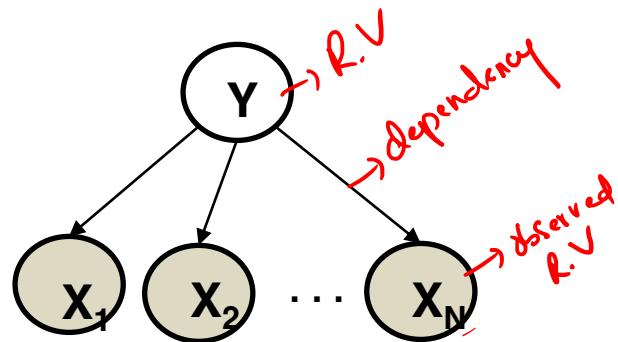
Topics-Words

# Latent Dirichlet Allocation



Length of the articles?  
Poisson distribution

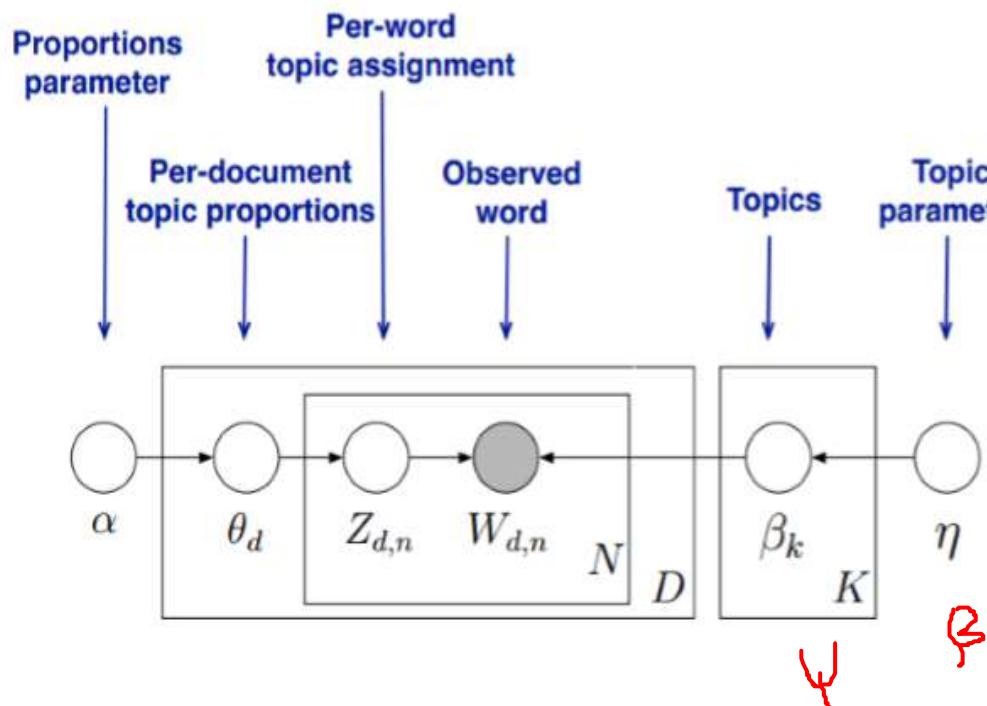
# Directed graphical model



$$P(Y, x_1, x_2, \dots, x_N) = P(Y) \prod_{n=1}^N P(x_n|y)$$

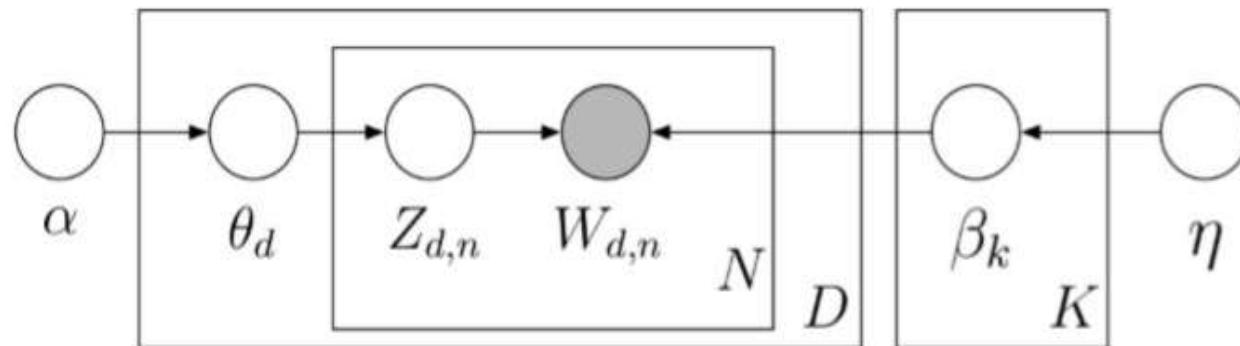
# Graphical LDA Model

Our goal is to **infer** or **estimate** the hidden variables, i.e. computing their distribution conditioned on the documents.  $\longrightarrow p(\text{topics, proportions, assignments} \mid \text{documents})$



- Nodes are RVs; edges indicate dependence.
- Shaded nodes are observed, and unshaded nodes are hidden.
- Plates indicate replicated variables.

# Graphical LDA Model



$K$  – total number of topics

$\beta_k$  – topic, a distribution over the vocabulary

$D$  – total number of documents

$\Theta_d$  – per-document topic proportions

$N$  – total number of words in a document (it fact, it should be  $N_d$ )

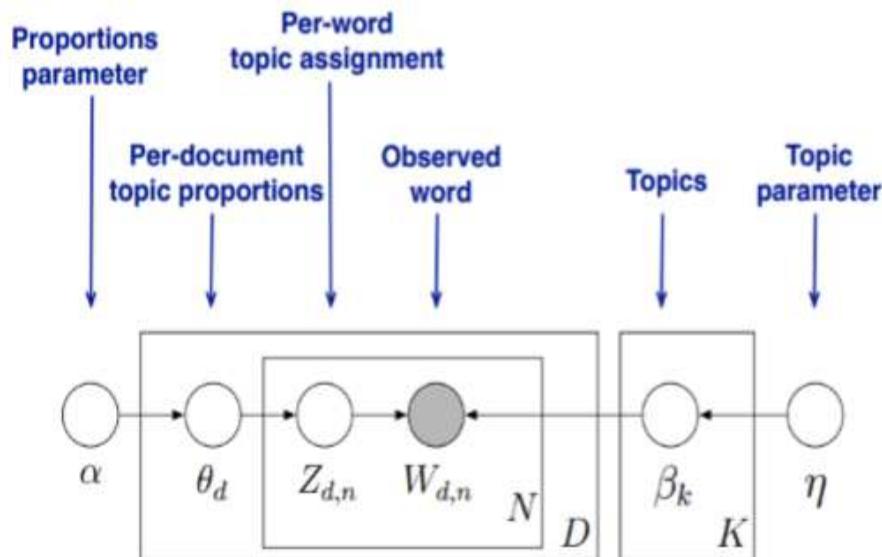
$Z_{d,n}$  – per-word topic assignment

$W_{d,n}$  – observed word

$\alpha, \eta$  – Dirichlet parameters

- Several **inference algorithms** are available (e.g. sampling based)
- A few **extensions** to LDA were created:
  - Bigram Topic Model

# Graphical LDA Model



1) Draw each topic  $\beta_i \sim Dir(\eta)$  for  $i = 1, \dots, K$

2) For each document:

First, Draw topic proportions  $\theta_d \sim Dir(\alpha)$

For each word within the document:

- Draw  $Z_{d,n} \sim Multi(\theta_d)$
- Draw  $W_{d,n} \sim Multi(\beta_{Z_{d,n}})$

$$p(\beta, \theta, z, w) = \left( \prod_{i=1}^K p(\beta_i | \eta) \right) \left( \prod_{d=1}^D p(\theta_d | \alpha) \prod_{n=1}^N p(z_{d,n} | \theta_d) p(w_{d,n} | \beta_{z_{d,n}}, z_{d,n}) \right)$$

# LDA Model

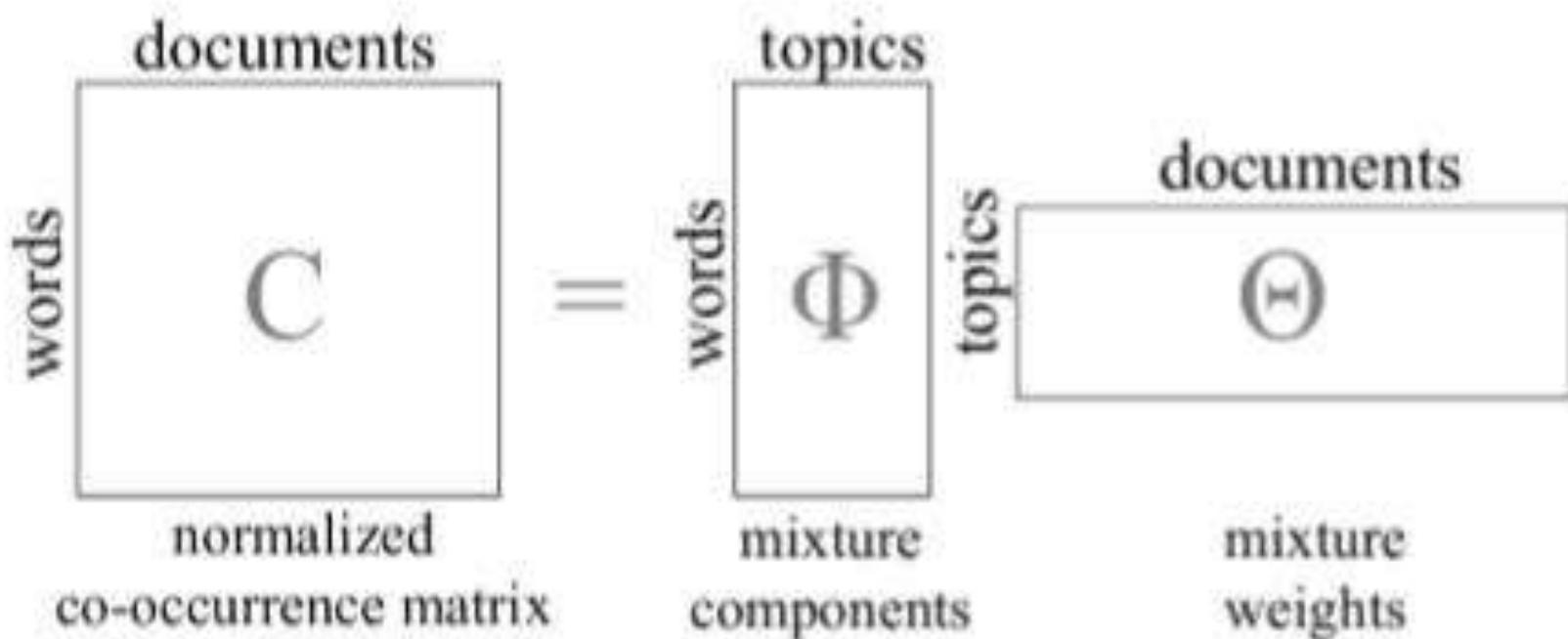
This joint distribution defines a posterior  $p(\theta, z, \beta | w)$ .

From a collection of documents we have to infer:

1. Per-word topic assignment  $z_{d,n}$ .
2. Per-document topic proportions  $\theta_d$ .
3. Per-corpus topic distributions  $\beta_k$ .

Then use posterior expectations ( $E\{\beta|w\}$  for the corpus,  $E\{\theta_d|w\}$  for each document) to perform the task at hand: information retrieval, document similarity, exploration, and others.

# LDA Matrix representation



# Mathematical modelling of LDA

Formal definition of the model:

$$p(\beta, \theta, z, w) = \left( \prod_{i=1}^K p(\beta_i | \eta) \right) \left( \prod_{d=1}^D p(\theta_d | \alpha) \prod_{n=1}^N p(z_{d,n} | \theta_d) p(w_{d,n} | \beta_{1:K}, z_{d,n}) \right)$$

$$(\beta_d | \eta) \sim Dir(\beta)$$

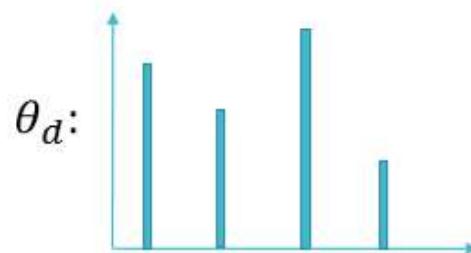
$$(\theta_d | \alpha) \sim Dir(\alpha)$$

$$Z_{d,n} \sim Multi(\theta_d)$$

$$W_{d,n} \sim Multi(\beta_{z_{d,n}})$$

$$p(z_{d,n} | \theta_d) = \theta_{d,z_{d,n}}$$

$$p(w_{d,n} | z_{d,n}, \beta_{1:K}) = \beta_{z_{d,n}, w_{d,n}}$$



$\beta:$

Topics	Word probabilities for each topic		
	Topic 1	Topic 2	Topic 3
Topic 1			
Topic 2			
Topic 3			
Topic 4			



# Train LDA - Gibbs Sampling



Sports



Politics



Science



Science

Science

Politics

Sports

# Assign Topics





Topic 1

Topic 2

Topic 3



Topic 1

Topic 2

Topic 3



80% Topic 3  
20% Topic 1

Topic 1



80% Topic 2  
20% Topic 1

Topic 2



80% Topic 1  
20% Topic 3

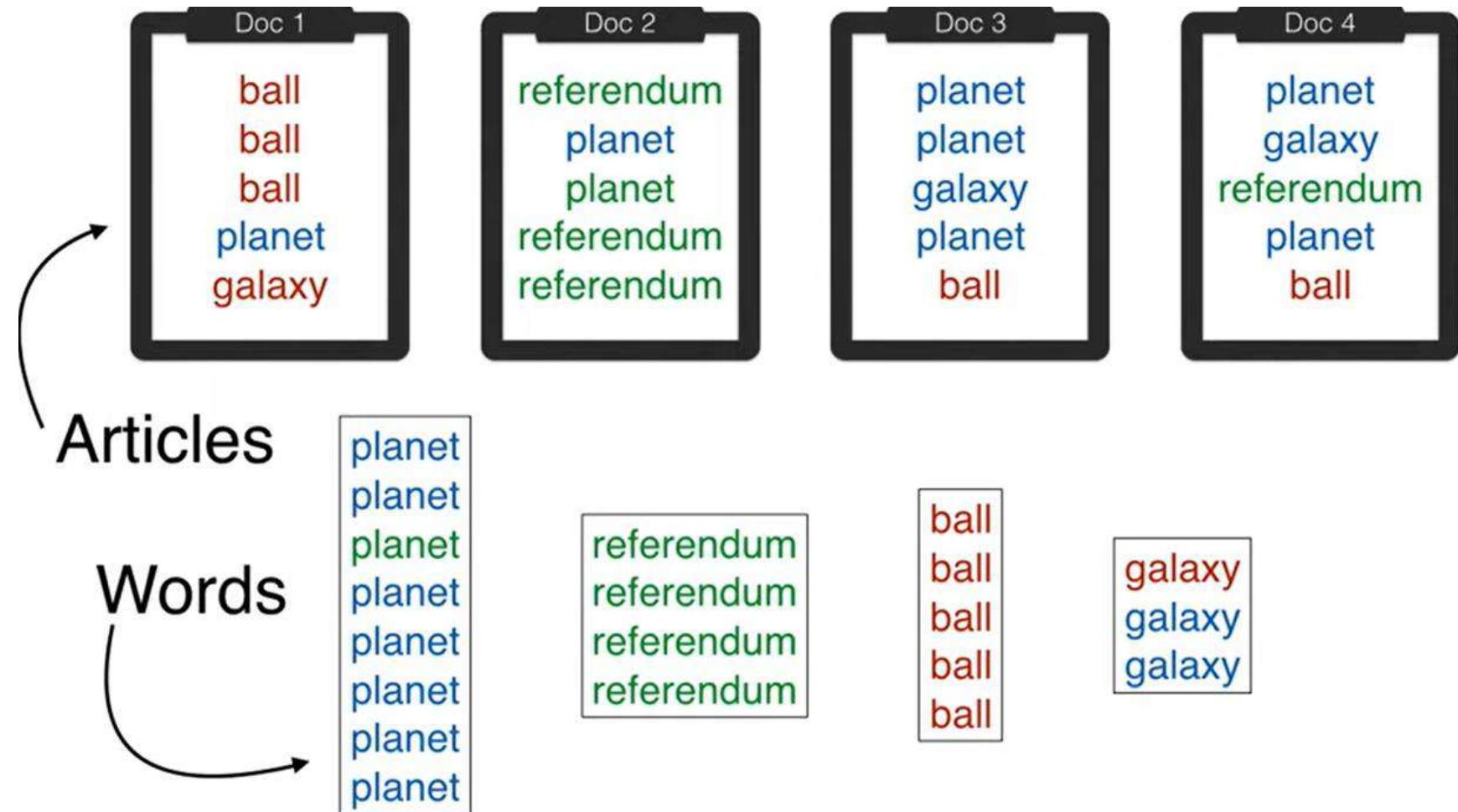
Topic 3



60% Topic 1  
20% Topic 2  
20% Topic 3



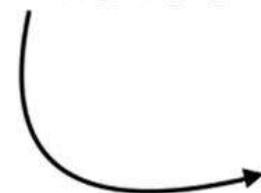
Property 1:  
Articles are as monochromatic as possible



---

## Property 2: Words are as monochromatic as possible

Words



planet  
planet  
planet  
planet  
planet  
planet  
planet  
planet

referendum  
referendum  
referendum  
referendum

ball  
ball  
ball  
ball  
ball

galaxy  
galaxy  
galaxy



Goal: Color each word with **blue**, **green**, **red**

1. Each article is as monochromatic as possible
2. Each word is as monochromatic as possible



Topic 1



Topic 2



ball

Topic 3

How much is Topic 1 in Doc 1?

2

How much is 'ball' in Topic 1?

0

Product: 0

How much is Topic 2 in Doc 1?

0

How much is 'ball' in Topic 2?

1

Product: 0

How much is Topic 3 in Doc 1?

2

How much is 'ball' in Topic 3?

3

Product: 6



### Topic 1

How much is Topic 1 in Doc 1?

$$2 + \alpha$$

How much is 'ball' in Topic 1?

$$0 + \beta$$

### Topic 2

How much is Topic 2 in Doc 1?

$$0 + \alpha$$

How much is 'ball' in Topic 2?

$$1 + \beta$$

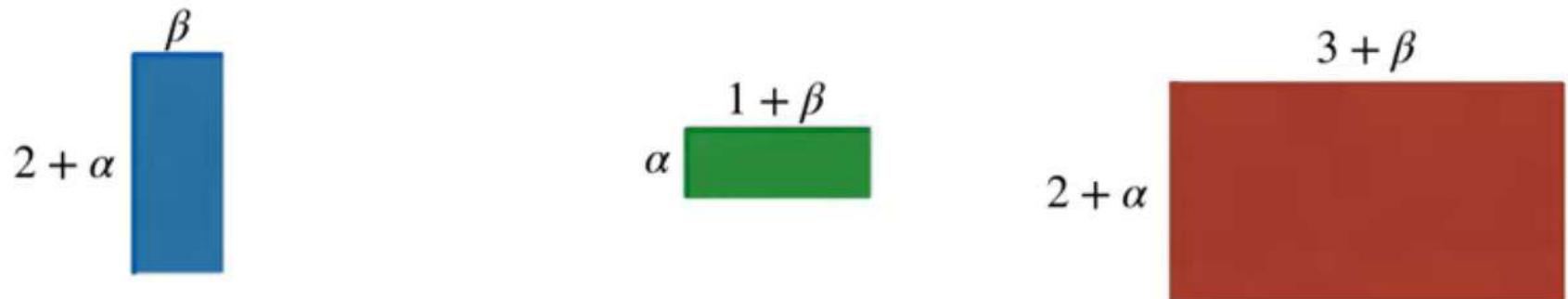
### Topic 3

How much is Topic 3 in Doc 1?

$$2 + \alpha$$

How much is 'ball' in Topic 3?

$$3 + \beta$$



### Topic 1

How much is Topic 1 in Doc 1?

$$2 + \alpha$$

How much is 'ball' in Topic 1?

$$0 + \beta$$

### Topic 2

How much is Topic 2 in Doc 1?

$$0 + \alpha$$

How much is 'ball' in Topic 2?

$$1 + \beta$$

### Topic 3

How much is Topic 3 in Doc 1?

$$2 + \alpha$$

How much is 'ball' in Topic 3?

$$3 + \beta$$



80% Topic 3  
20% Topic 1



80% Topic 2  
20% Topic 1



80% Topic 1  
20% Topic 3



60% Topic 1  
20% Topic 2  
20% Topic 3

### Science

Topic 1  
planet (7)  
galaxy (2)

### Politics

Topic 2  
referendum (4)  
planet (1)

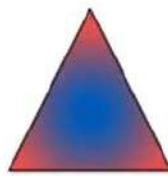
### Sports

Topic 3  
ball (5)  
galaxy (1)

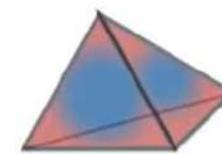
# Mathematical modelling of LDA

## Probability of a document

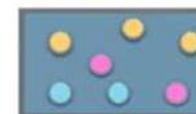
$$P(\mathbf{W}, \mathbf{Z}, \boldsymbol{\theta}, \boldsymbol{\varphi}; \alpha, \beta) = \prod_{j=1}^M P(\theta_j; \alpha) \prod_{i=1}^K P(\varphi_i; \beta) \prod_{t=1}^N P(Z_{j,t} | \theta_j) P(W_{j,t} | \varphi_{Z_{j,t}})$$



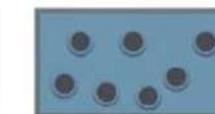
Topics



Words



Topics



Words

Dirichlet  
Distributions

Multinomial  
Distributions

## Probability of a document

## Gibbs sampling

# Gibbs Sampling Algorithm

- Step1: Assign a random topic [1...T] for each word
- Step2: For each word token, a new topic is sampled as per  $P(z_i=j|z_{-i}, w_i, d_i)$  and the matrices  $C_{wt}$  (word-topic) and  $C_{dt}$  (document-topic) are updated.
- One iteration over all word token in the document is a Gibbs Sample
- Each iteration may have correlation with the next hence these samples are saved at spaced intervals.

# LDA Summary

---

Documents are probability distributions over latent topics.

Topics are probability distributions over words.

LDA takes a number of documents. It assumes that the words in each document are related. It then tries to figure out the “recipe” for how each document could have been created. We just need to tell the model how many topics to construct and it uses that “recipe” to generate topic and word distributions over a corpus. Based on that output, we can identify similar documents within the corpus.

# LDA Summary

---

## ADVANTAGES

LDA is an effective tool for topic modeling.

Easy to understand conceptually

Has been shown to produce good results over many domains.

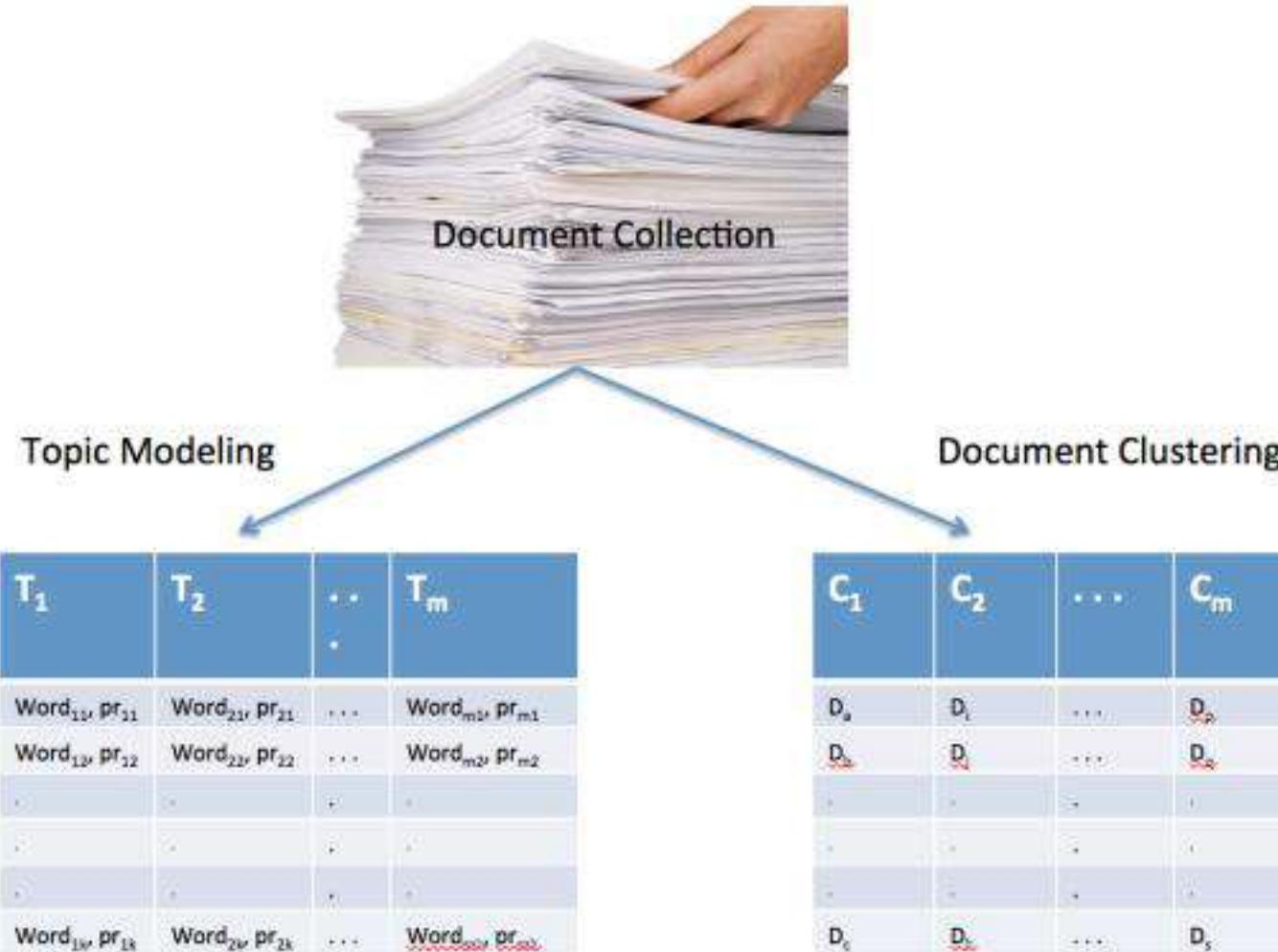
New applications

## LIMITATIONS

Must know the number of topics K in advance

Dirichlet topic distribution cannot capture correlations among topics

# Difference between document clustering and topic modeling



<https://iksinc.online/2016/05/16/topic-modeling-and-document-clustering-whats-the-difference/>

# Implementation Tools

## Tooling



**gensim:** topic modeling for humans

- Free python library
- Memory independent
- Distributed computing

<http://radimrehurek.com/gensim>



Stanford Topic Modeling Toolbox

<http://nlp.stanford.edu/software/tmt>



**MA**chine Learning for **LanguagE** Toolkit (MALLET) is a Java-based package for:

- statistical natural language processing
- document classification
- Clustering
- topic modeling
- information extraction
- and other machine learning applications to text.

<http://mallet.cs.umass.edu>

# References

---

Bernoulli trial, binomial and multinomial distribution:

<https://www.askiitians.com/iit-jee-algebra/probability/bernoulli-trials-and-binomial-distribution/>

Beta Distribution:

- [https://www.youtube.com/watch?v=v1uUgTcInQk&feature=emb\\_logo](https://www.youtube.com/watch?v=v1uUgTcInQk&feature=emb_logo)

Conjugate Prior:

- [https://www.youtube.com/watch?time\\_continue=2&v=aPNrhR0dFi8&feature=emb\\_logo](https://www.youtube.com/watch?time_continue=2&v=aPNrhR0dFi8&feature=emb_logo)
- <https://www.youtube.com/watch?v=qpNAxNmy0GU>

Topic Models

- <https://www.youtube.com/watch?v=fCmlceNqVog>

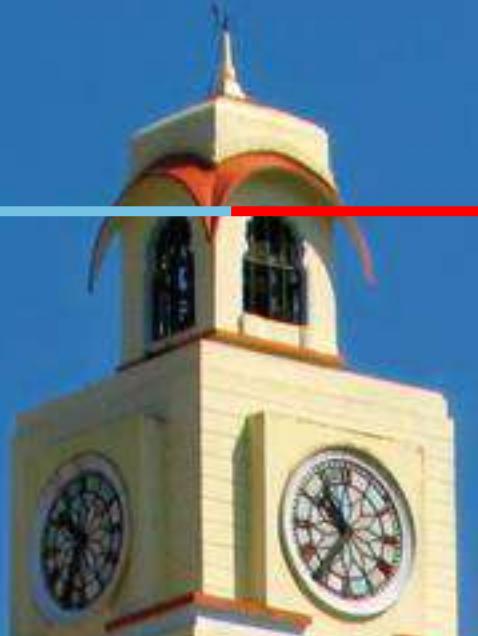
Gibbs Sampling

- <https://www.youtube.com/watch?v=u7l5hhmdc0M>
- <https://medium.com/analytics-vidhya/topic-modeling-using-lda-and-gibbs-sampling-explained-49d49b3d1045>

# References

## LDA

- [1] Blei, David M., Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation." Journal of machine Learning research (2003): 993-1022.
- <https://www.youtube.com/watch?v=3mHy4OSyRf0>
- <https://www.coursera.org/learn/text-mining/lecture/dmpQ0/2-5-topic-mining-and-analysis-motivation-and-task-definition>
- <https://www.youtube.com/watch?v=NYkbqzTIW3w>
- <https://github.com/adashofdata/nlp-in-python-tutorial>
- [https://www.youtube.com/watch?time\\_continue=3&v=Cpt97BpI-t4&feature=emb\\_logo](https://www.youtube.com/watch?time_continue=3&v=Cpt97BpI-t4&feature=emb_logo)
- <https://github.com/bhattbhavesh91>
- <https://www.youtube.com/watch?v=T05t-SqKArY>
- [https://www.youtube.com/watch?v=BaM1uiCpj\\_E](https://www.youtube.com/watch?v=BaM1uiCpj_E)
- <https://livebook.manning.com/book/grokking-machine-learning/>



# C5: Text Mining



**BITS Pilani**  
Hyderabad Campus

Dr. Chetana Gavankar, Ph.D,  
IIT Bombay-Monash University Australia  
[Chetana.gavankar@pilani.bits-pilani.ac.in](mailto:Chetana.gavankar@pilani.bits-pilani.ac.in)



**Session 4**  
**Date – 16<sup>th</sup> June 2024**  
**Time – 10 am to 12.15 pm**

These slides are prepared by the instructor, with grateful acknowledgement of Prof. Luis G. Serrano, Prof. Andrew Ng and many others who made their course materials freely available online.

# Outline

---

- Introduction to Sentiment Analysis
  - Sentiment Analysis
  - Subjectivity Analysis
  - Topic Extraction
  - Product Reviews
  - Opinion Retrieval and Spam
  - Opinion Summarization
  - Implementing Sentiment Analysis in Python
-

# Motivation For Sentiment Analysis



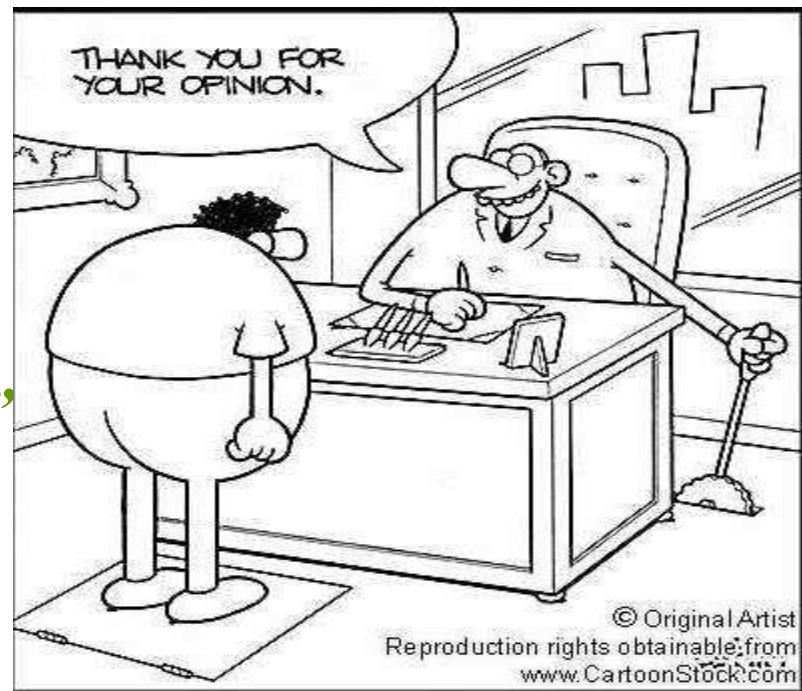
*What others think* has always been an important piece of information

*“Which car should I buy?”*

*“Which schools should I apply to?”*

*“Which Professor to work for?”*

*“Whom should I vote for?”*



# “So whom shall I ask?”

## Pre Web

- Friends and relatives
- Acquaintances
- Consumer Reports



## Post Web

*“...I don’t know who..but apparently it’s a good phone. It has good battery life and...”*

- Blogs (google blogs, livejournal)
- E-commerce sites (amazon, ebay)
- Review sites (CNET, PC Magazine)
- Discussion forums ([forums.craigslist.org](http://forums.craigslist.org),  
[forums.macrumors.com](http://forums.macrumors.com))
- Friends and Relatives (occasionally)



# The problem is..

---

- “Whoala! I have the reviews I need”
- *Now that I have “**too much**” information on one topic...I could easily form my opinion and make decisions...*

• Is this true?

• ...Not Quite

- Searching for reviews may be difficult
    - Can you search for opinions as conveniently as general Web search?
    - eg: is it easy to search for “*iPhone vs Google Phone*”?
-

# Facts and Opinions

---

Two main types of information on the Web.

- Facts(Objective) and Opinions(Subjective)

Fact : Thursday is a day.

Opinion : Thursday was a fun day.

Fact : iPhone is an Apple product.

Opinion : iPhone is good.

- Google searches for facts (currently)
  - Facts can be expressed with topic keywords
  - Google does not search for opinions
  - Opinions are hard to express with keywords
-

# Issues

---

- Not all subjective sentences contain opinions, e.g.
  - “*I want a phone with good voice quality*”
- Not all objective sentences contain no opinions, e.g.
  - “*The earphone broke in just two days!*”

# What is Sentiment analysis

- Computational study of opinions, sentiments, evaluations, attitudes, appraisal, affects, views, emotions, subjectivity, etc., expressed in text.



# Sentiment Analysis



It's a big day & I want to book a table at  
a nice Japanese restaurant



Seattle has many  
★★★★★  
sushi restaurants



What are people  
saying about  
the food?  
the ambiance?...



# Sentiment Analysis



Positive reviews not positive about everything

Sample review:

Watching the chefs create incredible edible art made the experience very unique.

My wife tried their ramen and it was pretty forgettable.

All the sushi was delicious! Easily best sushi in Seattle.

Experience



# Sentiment Analysis



## From reviews to topic sentiments

All reviews  
for restaurant

★★★★★ 1 review  
This is probably my favorite place in all Japanese in Seattle. My boyfriend and I ordered a lot of sushi, Japanese soups (miso), and the signature tsukiji and salmon rolls. I would say the salmon rolls, because the salmon was so fresh and the rice was perfectly cooked. We recommended by other Japanese were amazing. It's more chance and the consistency is the perfect amount of flavor for the delicate rolls.

★★★★★ 87 reviews  
Wrote here on the website last month and the writing front row to the dining room entrance. When I have more, I might do it again. I have never had better ramen here. I had two bowls of ramen, and got the set here today. The food was delicious!

★★★★★ 1 review  
I came here having high expectations due to the reviews of this place, but I was still disappointed. The service was terrible. A wait of an hour and a half for our table. When you come here, shrimp cost from \$4.00 each and shrimp can easily go up to \$10.00 each.

Novel intelligent  
restaurant review app

Experience  
★★★★★

Ramen  
★★★★

Sushi  
★★★★★

Easily best sushi  
in Seattle.

# Examples



The screenshot shows the top navigation bar of the Amazon.com website. It includes the Amazon logo with 'prime' underneath, a search bar with a magnifying glass icon, and various top-level links: Buy Again, Browsing History, Cody's Amazon.com, Early Black Friday Deals, Gift Cards, Registry, Sell, and Help.

LG Electronics OLED55E8PUA 55-Inch 4K Ultra HD Smart OLED TV (2018... > Customer reviews

## Customer reviews

★★★★★ 18

3.9 out of 5 stars >



## LG Electronics OLED55E8PUA 55-Inch 4K Ultra HD Smart OLED TV (2018 Model)

by LG

Size: 55-inch Change

Price: \$2,296.99 

[Write a review](#)

### Top positive review

[See all 14 positive reviews >](#)



Mayra S. TOP 1000 REVIEWER

★★★★★ With Google Assistant and new Alpha 9 Processor, 2018 LG Oled's are great upgrades for first time 4K/HDR/Oled Owners

May 3, 2018

### Top critical review

[See all 4 critical reviews >](#)



Brett W.

★★★★☆ Extreme stuttering (no soft transition between frames) is an important factor to consider with OLED TV's

August 3, 2018

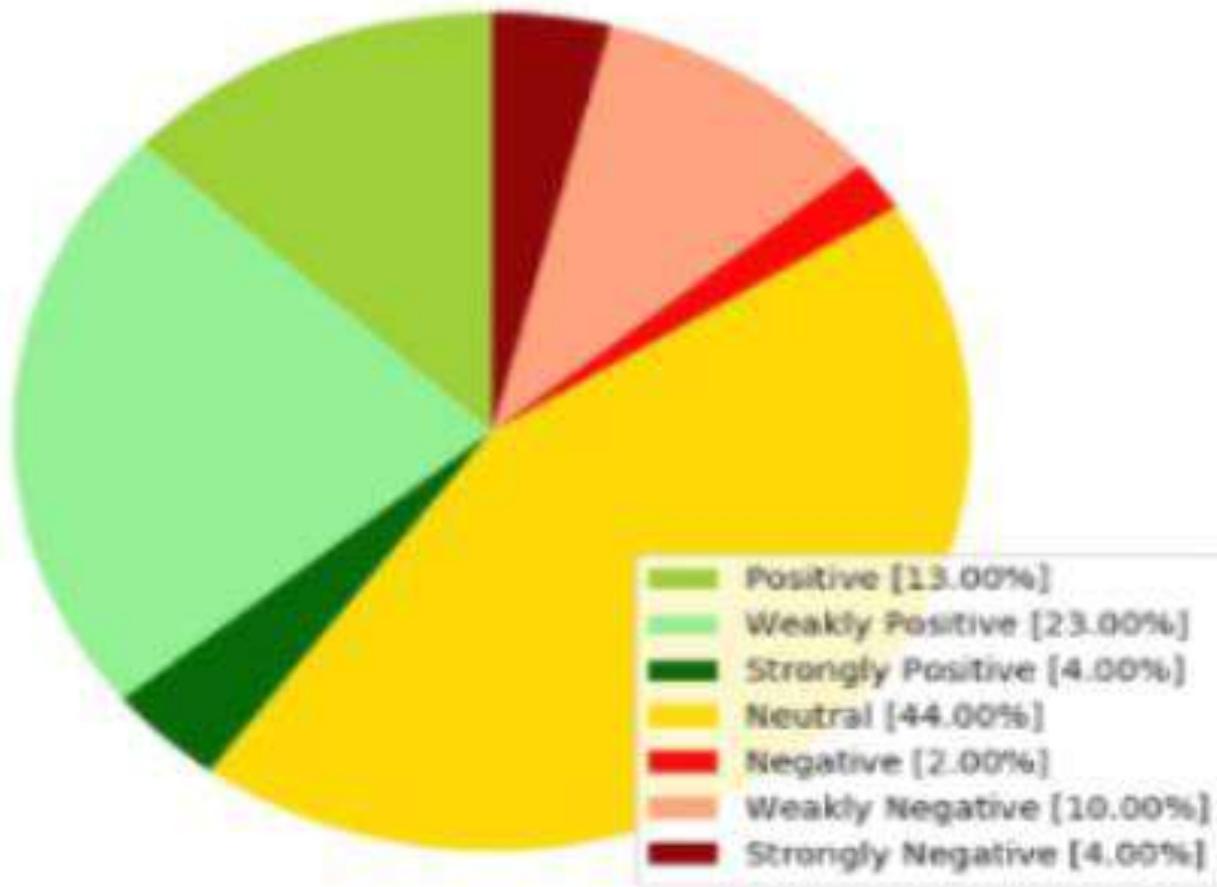
# Examples

*Data gathered from the analysis of +4,000 product reviews*



# Examples

How people are reacting on bitcoin by analyzing 100 Tweets.



# Examples

## Sentiment Analytics for the Telecom Company

### Negative review

Dear #XYZ there is no network in my area and internet service is pathetic from the past one week. Kindly help me out

-Dated: 10/09/17

### Mixed review

Although the value added services being provided are great but the prices are high  
#VAS #XYZ

-Dated: 10/09/17

### Positive review

Great work done #XYZ Problem resolved by customer care in just one day  
#ThanksXYZ

-Dated: 5/06/17



Sentiment Analytics Model

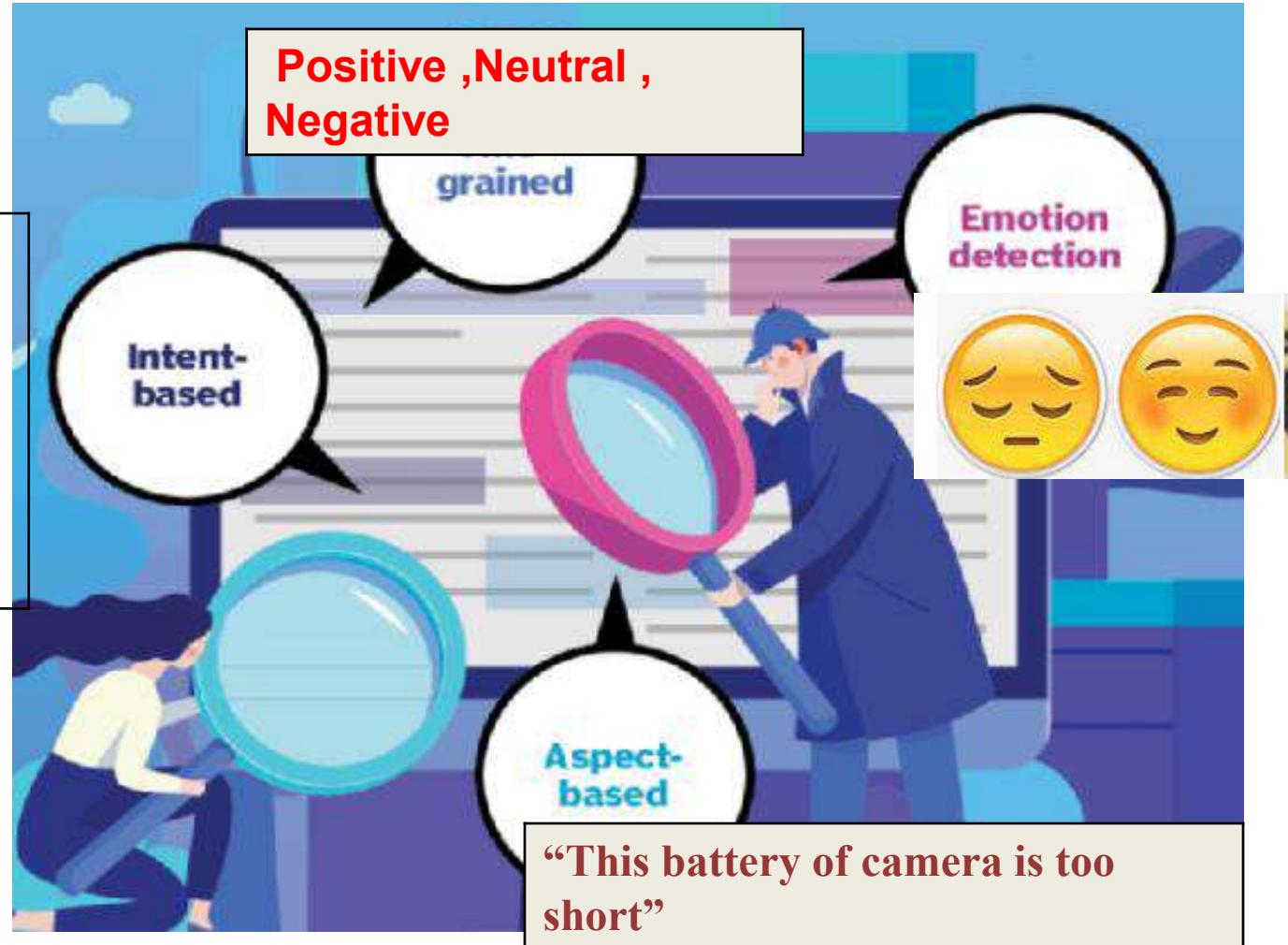
Analyze output to implement **Actionables** like:

- Improve service quality and increase quality checks for locations with max negative reviews.
- Train employees and improve infrastructure of areas with poor service.
- Ensure dedicated customer complaint teams for some areas.
- Tweak the marketing strategy to convey the right message to the customers.

\* XYZ depicts name of the telecom company

# Types of Sentiment Analysis

intention to sell,  
intention to complain or intention to purchase etc





# Different levels of sentiment analysis

---

- Three levels of granularity
    - Document level
    - Sentence level
    - Entity and Feature/Aspect level
-

# Sentiment analysis methods

---

- **Rule-based** systems that perform sentiment analysis based on a set of manually crafted rules.
  - **Automatic** systems that rely on machine learning techniques to learn from data.
  - **Hybrid** systems that combine both rule-based and automatic approaches.
-

# Rule based methods

---

Following steps need to be performed

- Extract the data
  - Tokenize text. The task of splitting the text into individual words
  - Stop words removal. Those words which do not carry any significant meaning and should not be used for the analysis activity. Examples of stop words are: a, an, the, they, while etc.
  - Punctuation removal (in some cases)
  - Running the *preprocessed* text against the sentiment lexicon which should provide the number/measurement corresponding to the inferred emotion
-

# Example

“Sam is a great guy”

## 1. Tokenize



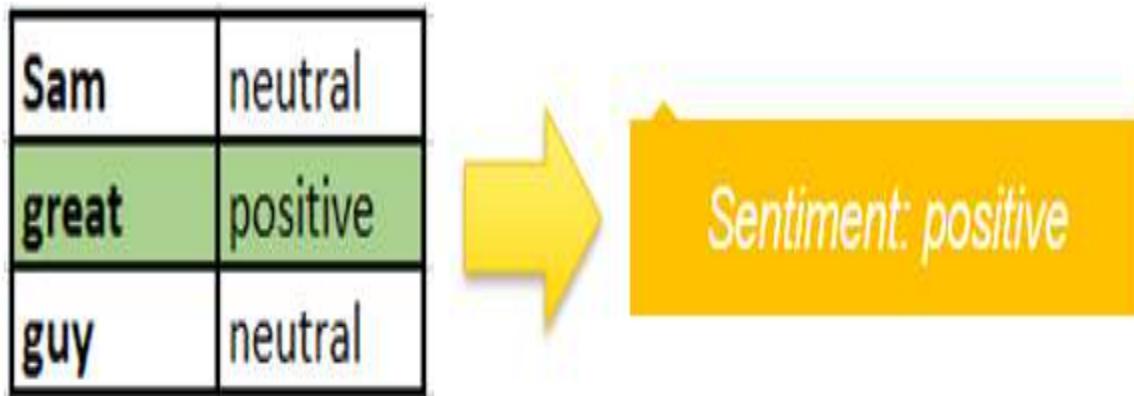
## 2. Remove stop words and punctuations

Tokenize	Preprocessing
Sam	
is	stop word
a	stop word
great	
guy	
.	punctuation



Sam	neutral
great	positive
guy	neutral

- 
3. Running the lexicon on the preprocessed data, returns a **positive sentiment** score/measurement because of the presence of a positive word “great” in the input data.



# Machine learning Approach

*The song was good .*

## 1.Tokenization

- The
- Song
- Was
- Good
- .

## 2.Cleaning the data (Remove special characters )

- The
- Song
- Was
- Good

# Contd..

---

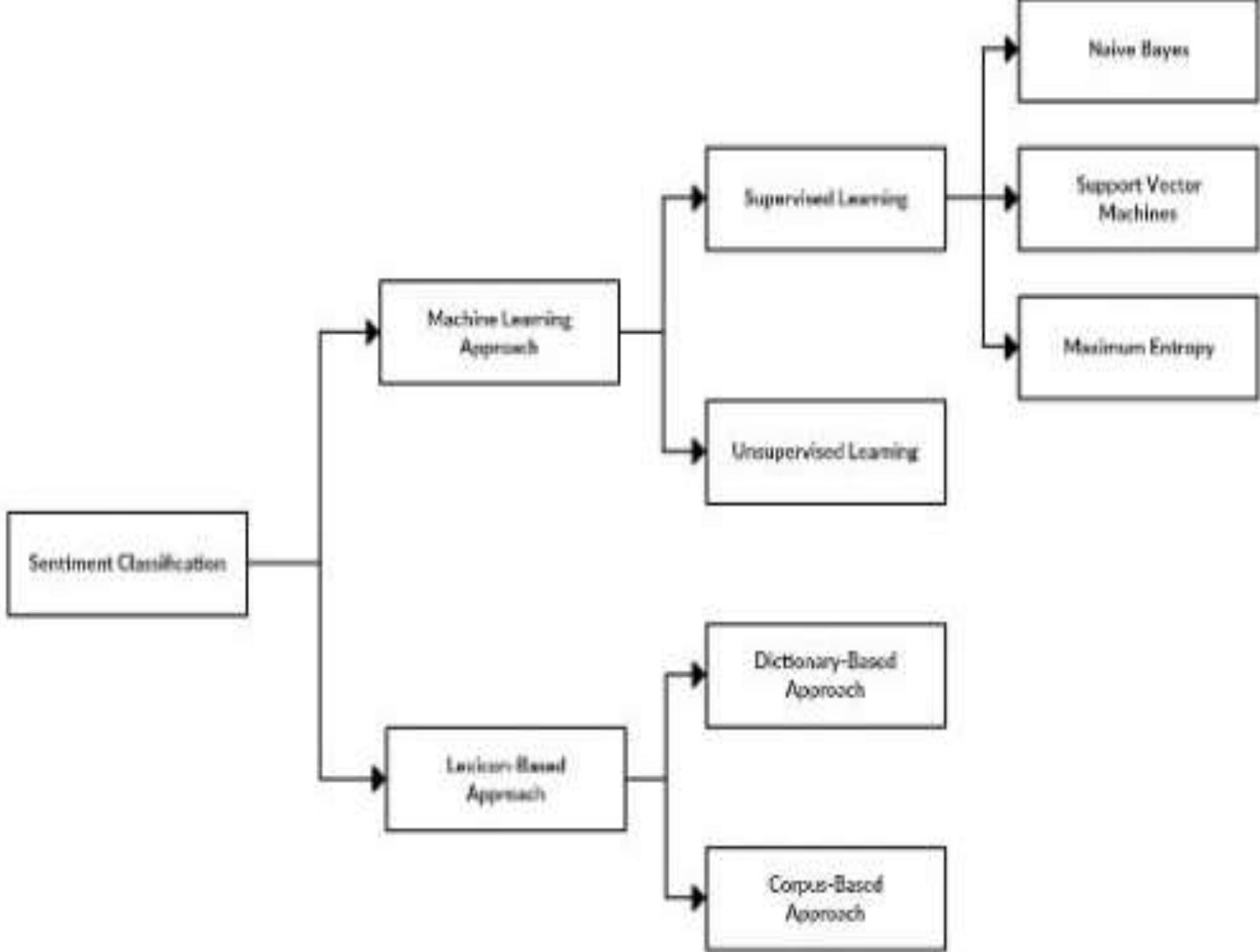
## 3. Remove stop words.

- Song
- good

## 4. Classification(Positive, negative ,Neutral)

Apply supervised algorithm

- *Naïve Bayes*
  - *Support vector machines*
  - *Maximum Entropy*
-



# Sentiments/ Emotions

---

- Most models include the two dimensions valence and arousal, and many add a third, dominance.
  - These can be defined as:
    - valence: the pleasantness of the stimulus
    - arousal: the intensity of emotion provoked by the stimulus
    - dominance: the degree of control exerted by the stimulus
-

# Lexicons

---

- Many sentiment applications rely on lexicons to supply features to a model.
  - A lexicon is a **resource with information about words**.
  - A sentiment lexicon has information such as list of words which are positive and negative.
-

# General Inquirer

- Harvard General Inquirer Database (Stone, 1966)
  - Total of 11,788 terms
  - [http://www.wjh.harvard.edu/~inquirer/spreadsheet\\_guide.htm](http://www.wjh.harvard.edu/~inquirer/spreadsheet_guide.htm)
  - <http://www.wjh.harvard.edu/~inquirer/homecat.htm>
  - Positive (1915 words) vs Negative (2291 words)
  - Strong vs Weak
  - Active vs Passive
  - Overstated versus Understated
  - Pleasure, Pain, Virtue, Vice
  - Motivation, Cognitive Orientation, etc

# Sample

A fragment of the Harvard General Inquirer spreadsheet file.

Entry	Positiv	Negativ	Hostile	...184 classes ...	Othtags	Defined
1	A				DET ART	...
2	ABANDON	Negativ			SUPV	
3	ABANDONMENT	Negativ			Noun	
4	ABATE	Negativ			SUPV	
5	ABATEMENT				Noun	
...						
35	ABSENT#1	Negativ			Modif	
36	ABSENT#2				SUPV	
...						
11788	ZONE				Noun	



# SentiWordNet

- Home page :<http://sentiwordnet.isti.cnr.it/>
- All WordNet synsets automatically annotated for degrees of positivity ,negativity and neutrality/objectiveness.

# Example

POS	ID	PosScore	NegScore	SynsetTerms	Gloss
a	00001740	0.125	0	able#1	(usually followed by 'to') having the necessary means or [...]
a	00002098	0	0.75	unable#1	(usually followed by 'to') not having the necessary means or [...]
a	00002312	0	0	dorsal#2 abaxial#1	facing away from the axis of an organ or organism; [...]
a	00002527	0	0	ventral#2 adaxial#1	nearest to or facing toward the axis of an organ or organism; [...]
a	00002730	0	0	acrosopic#1	facing or on the side toward the apex
a	00002843	0	0	basiscopic#1	facing or on the side toward the base
a	00002956	0	0	abducting#1 abducent#1	especially of muscles; [...]
a	00003131	0	0	adductive#1 adducting#1 adducent#1	especially of muscles; [...]
a	00003356	0	0	nascent#1	being born or beginning; [...]
a	00003553	0	0	emerging#2 emergent#2	coming into existence; [...]



# MPQA Subjectivity Cues Lexicon

- <https://mpqa.cs.pitt.edu/>
- 6885 words from 8221 lemmas
  - 2718 positive
  - 4912 negative
- Each word annotated for intensity (strong, weak)

	<b>Strength</b>	<b>Length</b>	<b>Word</b>	<b>Part-of-speech</b>	<b>Stemmed</b>	<b>Polarity</b>
1.	type=weaksubj	len=1	word1=abandoned	pos1=adj	stemmed1=n	priorpolarity=negative
2.	type=weaksubj	len=1	word1=abandonment	pos1=noun	stemmed1=n	priorpolarity=negative
3.	type=weaksubj	len=1	word1=abandon	pos1=verb	stemmed1=y	priorpolarity=negative
4.	type=strongsubj	len=1	word1=abase	pos1=verb	stemmed1=y	priorpolarity=negative
5.	type=strongsubj	len=1	word1=abasement	pos1=anypos	stemmed1=y	priorpolarity=negative
6.	type=strongsubj	len=1	word1=abash	pos1=verb	stemmed1=y	priorpolarity=negative
7.	type=weaksubj	len=1	word1=abate	pos1=verb	stemmed1=y	priorpolarity=negative
8.	type=weaksubj	len=1	word1=abdicate	pos1=verb	stemmed1=y	priorpolarity=negative
9.	type=strongsubj	len=1	word1=aberration	pos1=adj	stemmed1=n	priorpolarity=negative
10.	type=strongsubj	len=1	word1=aberration	pos1=noun	stemmed1=n	priorpolarity=negative
...						
8221.	type=strongsubj	len=1	word1=zest	pos1=noun	stemmed1=n	priorpolarity=positive

# Linguistic inquiry and word count

---

- Home Page: <http://www.liwc.net/>
  - 2300 word > 70 classes
  - Affective Processes
  - Negative emotion (bad, weird, hate, problem,tough)
  - Positive emotion (love,nice,sweet)
  - Cognitive Processes
-

Category	Examples
Negate	aint, ain't, arent, aren't, cannot, cant, can't, couldnt, ...
Swear	arse, arsehole*, arses, ass, asses, asshole*, bastard*, ...
Social	acquainta*, admit, admits, admitted, admitting, adult, adults, advice, advis*
Affect	abandon*, abuse*, abusi*, accept, accepta*, accepted, accepting, accepts, ache*
Posemo	accept, accepta*, accepted, accepting, accepts, active*, admir*, ador*, advantag*
Negemo	abandon*, abuse*, abusi*, ache*, aching, advers*, afraid, aggravat*, aggress*,
Anx	afraid, alarm*, anguish*, anxi*, apprehens*, ashame*, aversi*, avoid*, awkward*
Anger	jealous*, jerk, jerked, jerks, kill*, liar*, lied, lies, lous*, ludicrous*, lying, mad

# Bing Liu Opinion Lexicon

---

- [Bing Liu's Page on Opinion Mining](#)
  - <http://www.cs.uic.edu/~liub/FBS/opinion-lexicon-English.rar>
  - 6786 words
    - 2006 positive
    - 4783 negative
-

# Corpus based lexicon generator

---

- A more sophisticated technique is a corpus-based approach which relies on syntactic or **co-occurrence patterns** together with a seed list of opinion words.
  - The technique **starts with a list of seed opinion adjective words**, and uses them and a **set of linguistic constraints** or conventions on connectives to **identify additional adjective opinion words and their orientations**.
-

# Bootstrapping architecture



# Example

---

- Adjectives conjoined by “and” have same polarity

*Fair and legitimate ,corrupt and brutal*

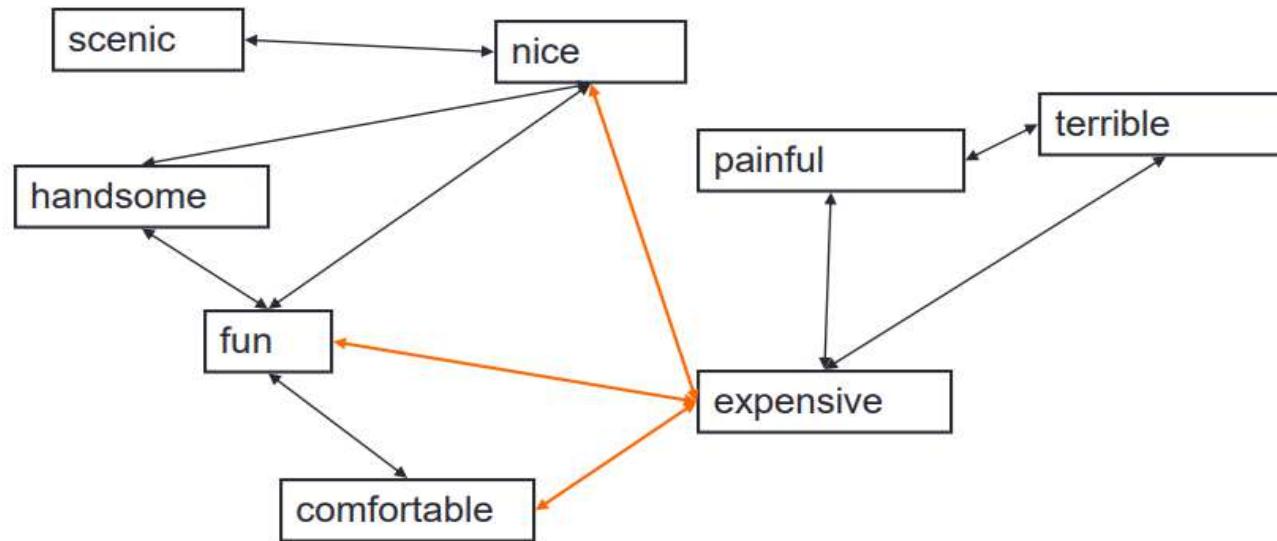
- Adjectives conjoined by “but” do not

*Fair but brutal*



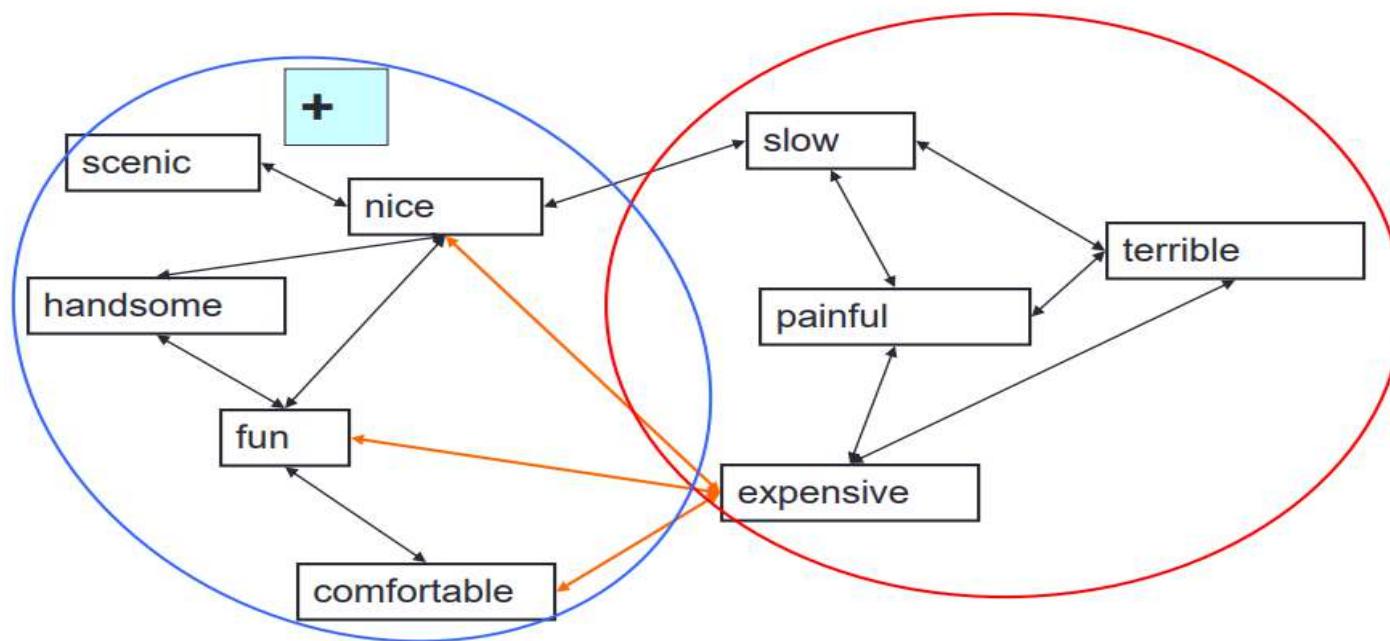
# Algorithm

1. Generate a Labeled seed set of adjectives
2. Expand seed set to conjoined adjectives by looking up in a corpus/web search
3. builds a graph of adjectives linked by the same or different semantic orientation



# Clustering Algorithm

- A clustering algorithm partitions the adjectives into two subsets



# Turney algorithm

- Extract a phrasal lexicon from reviews
- Learn polarity of each phrase
- Rate a review by the average polarity of its phrases

First Word	Second Word	Third Word (not extracted)
JJ	NN or NNS	anything
RB, RBR, RBS	JJ	Not NN nor NNS
JJ	JJ	Not NN or NNS
NN or NNS	JJ	Not NN nor NNS
RB, RBR, or RBS	VB, VBD, VBN, VBG	anything

Two-word phrases with adjectives

# How to measure polarity of a phrase

---

- Positive phrases co-occur more with “excellent”
  - Negative phrases co-occur more with “poor”
  - But how to measure co-occurrence?
-

# Pointwise Mutual Information

---

- Pointwise mutual information: How much more do events x and y co-occur than if they were independent?

$$\text{PMI}(word_1, word_2) = \log_2 \frac{P(word_1, word_2)}{P(word_1)P(word_2)}$$

- If two words tend to not at all **co-occur** , PMI is negative or zero
  - If two words tend to co-occur , **PMI is positive**
  - Does phrase appear more with “poor” or “excellent”?  
–Polarity(phrase) = PMI(phrase, "excellent") – PMI(phrase, "poor")
-

# Two reviews for Positive and Negative phrases



Phrase	POS tags	Polarity
online service	JJ NN	2.8
online experience	JJ NN	2.3
direct deposit	JJ NN	1.3
local branch	JJ NN	0.42
...		
low fees	JJ NNS	0.33
true service	JJ NN	-0.73
other bank	JJ NN	-0.85
inconveniently located	JJ NN	-1.5
<i>Average</i>		0.32

Phrase	POS tags	Polarity
direct deposits	JJ NNS	5.8
online web	JJ NN	1.9
very handy	RB JJ	1.4
...		
virtual monopoly	JJ NN	-2.0
lesser evil	RBR JJ	-2.3
other problems	JJ NNS	-2.8
low funds	JJ NNS	-6.8
unethical practices	JJ NNS	-8.5
<i>Average</i>		-1.2

# Wordnet based polarity estimation

---

- WordNet: online thesaurus indexing words by synonyms
  - Create positive (“good”) and negative seed-words (“terrible”)
  - Find Synonyms and Antonyms
    - Positive Set: Add synonyms of positive words (“well”) and antonyms of negative words
    - Negative Set: Add synonyms of negative words (“awful”) and antonyms of positive words (“evil”)
  - Repeat, following chains of synonyms
  - Filter
-



# Aspect Based Sentiment Analysis (ABSA)

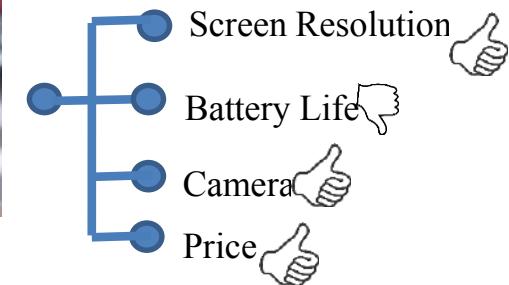
---

“(1) I bought an *iPhone* a few days ago. (2) It was such a *nice phone*. (3) The *touch screen* was really *cool*. (4) The *voice quality* was *clear* too. (5) Although the *battery life* was *not long*, that is ok for me. (6) However, *my mother* was mad with me as I did not tell her before I bought it. (7) She also thought the *phone* was too *expensive*, and wanted me to return it to the shop. . . ”

# Aspect Based Sentiment Analysis (ABSA)

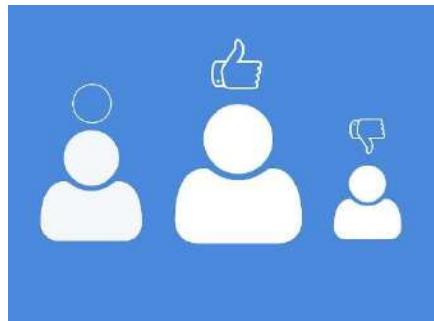


- Each opinion is defined as **quintuple**  $(e, a, s, h, t)$ , where  $e$  is an entity and  $a$  is one of its aspects,  $s$  is the sentiment on the aspect  $a$ ,  $h$  is the opinion holder and  $t$  is the time when the opinion is expressed.
- Find the **target(Aspect/Entity)** of the sentiment.
- Two approaches
  - Find most common noun phrases
  - Build a classifier



# Frequency-Based Aspect Extraction

- A key characteristic is that an **opinion always has a target**.
- Exploit **syntactic structures** to depict opinion and target relationships



Review corpus



Association Rule  
Mining

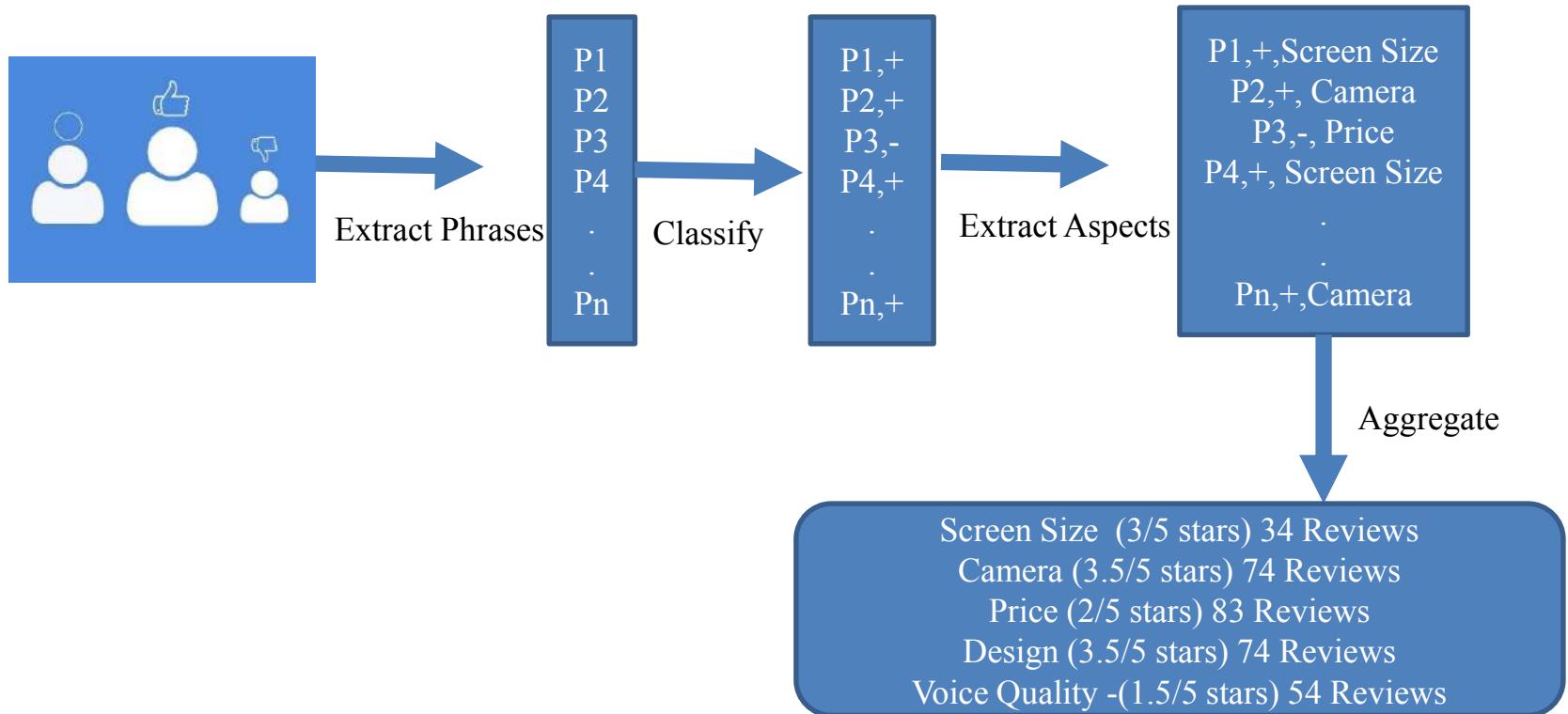
Screen Size – 100/500  
Camera Resolution – 300/500  
Battery Life – 350/500  
Price -450/500  
Voice clarity – 325/500

# Examples of aspects extracted

- Those candidate aspects with the highest frequency counts are almost always the most important aspects of the product.
- Assumption: Corpus has reasonable number of reviews and belong to same product.

Entity	Aspects extracted
Casino	Casino, buffet, pool, resort, beds
Department store	Selection, department, sales, shop, clothing
Greek Restaurant	Food, Wine, Service, Appetizer, lamb

# Architecture for ABSA



Blair-Glodensohn from Google

# How to deal with star ratings?

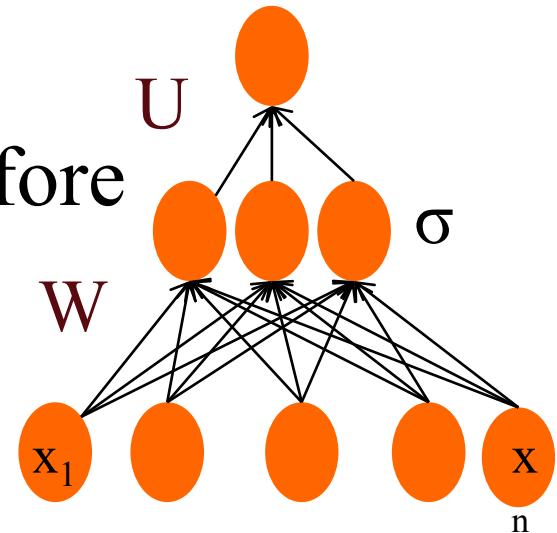
---

- Binarization of the star ratings
- Use regression instead of a binary classifier.



# ML/DL algorithms

- We could do exactly what we did with logistic regression
- Input layer are binary features as before
- Output layer is 0 or 1

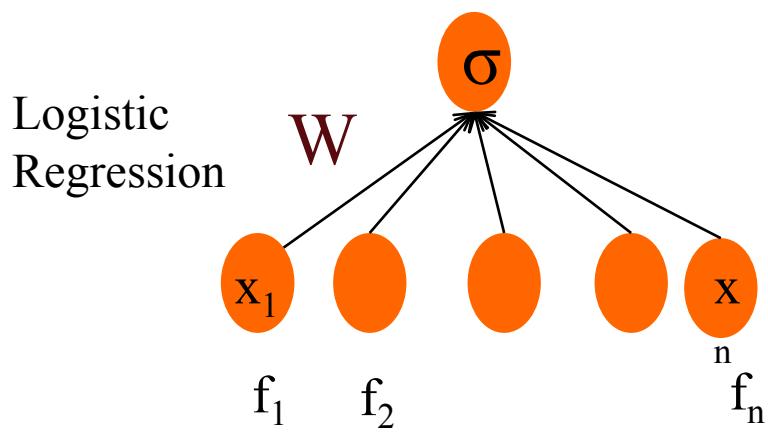


# Sentiment Features

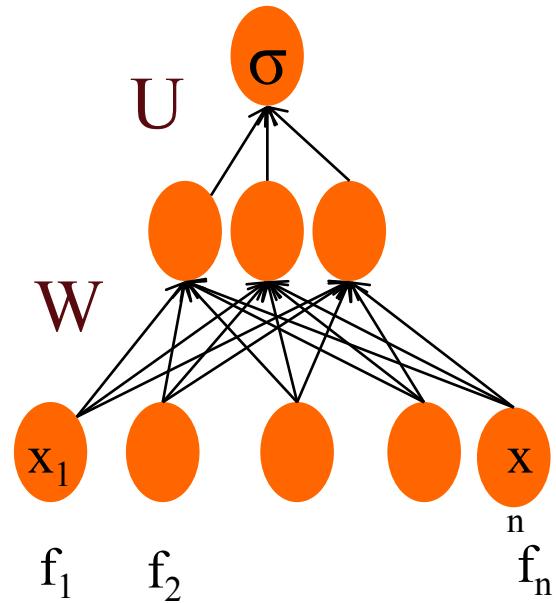
---

Var	Definition
$x_1$	count(positive lexicon) $\in$ doc)
$x_2$	count(negative lexicon) $\in$ doc)
$x_3$	$\begin{cases} 1 & \text{if “no”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$
$x_4$	count(1st and 2nd pronouns $\in$ doc)
$x_5$	$\begin{cases} 1 & \text{if “!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$
$x_6$	log(word count of doc)

# Feedforward nets for simple classification



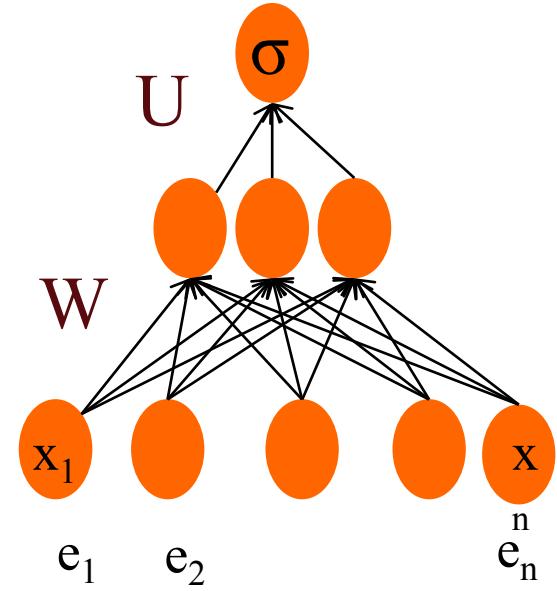
2-layer  
feedforward  
network



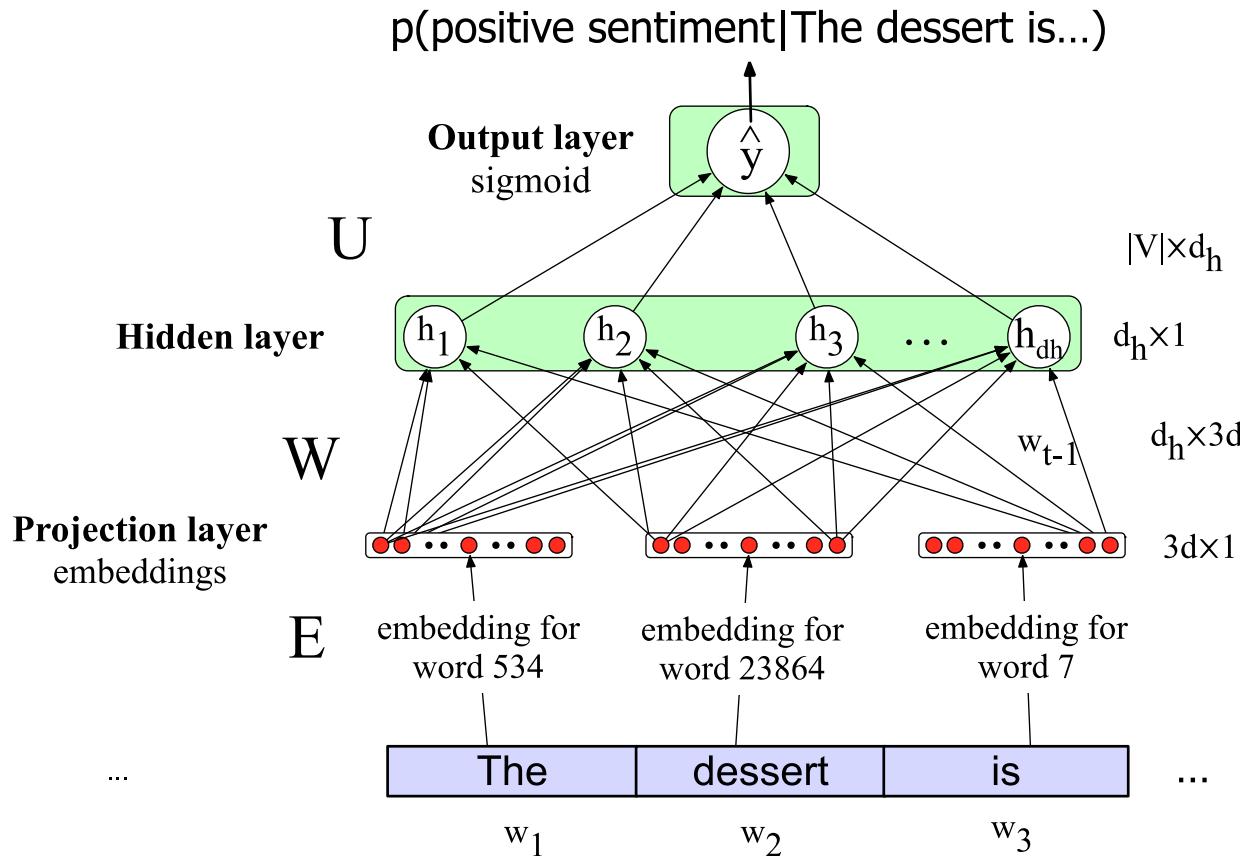
- Just adding a hidden layer to logistic regression
- allows the network to use non-linear interactions between features
- which may (or may not) improve performance.

# Even better: representation learning

- The real power of deep learning comes from the ability to **learn** features from the data
- Instead of using hand-built human-engineered features for classification
- Use learned representations like embeddings!



# Neural Net Classification with embeddings as input features!



# Opinion Spamming

---

- Types of Spam
    - Type 1 (fake reviews)
    - Type 2 (reviews about brands only)
    - Type 3 (non-reviews)
-

# Types of Data, Features and Detection

---

- Three main types of data have been used for review spam detection:
    - Review content
    - Meta-data about the review
    - Product information
-

# “Let me look at reviews on one site only...”

## Problems?



- Biased views
  - all reviewers on one site may have the same opinion
- Fake reviews/Spam (sites like YellowPages, CitySearch are prone to this)
  - people post good reviews about their own product OR services
  - some posts are plain spams

# Coincidence or Fake?



## Reviews for a moving company from YellowPages

- # of merchants reviewed by the each of these reviewers → 1
- Review dates close to one another
- All rated 5 star
- Reviewers seem to know exact names of people working in the company and TOO many positive mentions

**THE BEST!!!!** 11/30/2007 Posted by c\_karen ★★★★★

NorthStar did an outstanding job of packing and moving my things. Quite frankly I was expecting some things to be broken. However, to my surprise not one thing was broken and everything went as smooth as could be expected. I had approximately 15,000 lbs. of items to move. I am very impressed with NorthStar and I would not hesitate to utilize them again for my next move. All of the young men who assisted in packing and loading were very hard working and polite

**Pros:** everything was great

**GOOD MOVING** 10/11/2007 Posted by joanlee777 ★★★★★

About a month ago, on Sep 12, we hired NorthStar Moving to move our belongings from our house in Van Nuys to the Highway Storage place in Santa Clara. We would like to express our sincere thanks and appreciation for the professional work that was carried out by NorthStar team of workers. In particular, we would like to mention the four NorthStar workers: Roy Ashual, Moshiko Haziza, Guillermo Molise and Roberto Mendoza for their very dedicated service. Besides being good natured and helpful, they worked very well and took good care of our personal effects. We would definitely refer them and NorthStar Moving to any of our friends who are looking for a good moving company.

**Great movers** 10/08/2007 Posted by shelly\_morgan ★★★★★

I wanted to thank the Northstar Moving group for a fabulous job. We hired Northstar Moving on August 4th to move us out of two storage units and where we were staying to our new home in Los Angeles. I had gone through surgery on the 2nd and was in no condition to move around a lot. The Northstar Moving team was great. I slept in while my husband met them at the first pick-up point. Then they came to the 2nd and that is where I met them. When we arrived at the new house they found something for me to sit on and I sat in one place in the garage telling them which room the items went. They were great. They had wonderful personalities. I have never had so much fun moving (even if I was in some pain). Northstar thank you again for the great team and customer service.

# Supervised Spam Detection

---

- Opinion spam detection can be formulated as a classification problem with two classes, fake and non-fake.
  - Due to the fact that there is no labeled training data for learning, **Jindal and Liu (2008)** exploited duplicate reviews.
  - In their study of 5.8 million reviews and 2.14 million reviewers from amazon.com, a large number of duplicate and near-duplicate reviews were found.
-



# Four categories to handle duplicates and near duplicates

---

- Duplicates from the same user-id on the **same** product
  - Duplicates from **different** user-ids on the **same** product
  - Duplicates from the **same** user-id on **different** products
  - Duplicates from **different** user-ids on **different** products
-

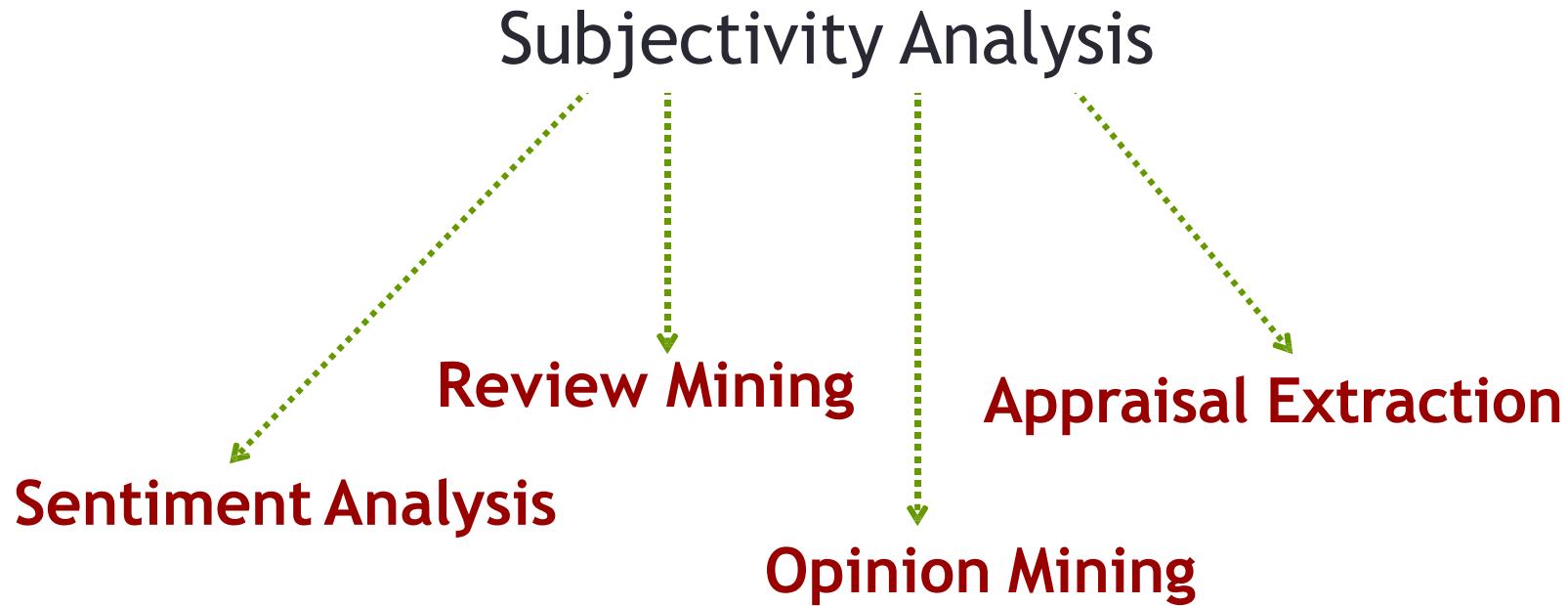


# Feature engineering for fake reviews

---

- Review centric features
  - Reviewer centric features
  - Product centric features
-

# Problem Names



Synonymous  
&  
Interchangeably Used!

# So, what is Subjectivity?

---



- The **linguistic** expression of somebody's **opinions**, **sentiments**, **emotions**.....(private states)
  - private state: state that is not open to objective verification (*Quirk, Greenbaum, Leech, Svartvik (1985). A Comprehensive Grammar of the English Language.*)
  - **Subjectivity analysis** - is the computational study of **affect**, **opinions**, and **sentiments** expressed in text
    - blogs
    - editorials
    - reviews (of products, movies, books, etc.)
    - newspaper articles
-

# Example: iPhone



## Lab test: Apple gets iPhone 3G right for business

An abundance of new features carries iPhone 3G and iPhone 2.0 into the enterprise

By Tom Yager  
July 24, 2008

Talkback E-mail Printer Friendly Reprints Text Size A A

### InfoWorld

- summary is structured
- everything else is plain text
- mixture of objective and subjective information
- no separation between positives and negatives

With the review iPhone 3G their te and You

Related

New M extra S

AT&T s tetheri

Popula apple,

See A

iPhone delivers more misses than hits

iPhone: The \$1,975 iPod

» Back to special report:  
Apple launches the iPhone 3G

#### The Bottom Line

#### Apple iPhone 3G

Apple, [apple.com/iphone](http://apple.com/iphone)

#### Very Good 8.5

criteria score weight

Extensibility 7 20%

Messaging 8 20%

Networking 9 20%

Usability 9 20%

Multimedia 10 10%

Value

Review on InfoWorld -  
tech news site

s for the device, a  
e 3G and the new  
among other things,  
an a cellular browser

#### Product summary

##### Good:

Apple iPhone has a stunning display, a sleek  
and an innovative multitouch user interface.  
browser makes for a superb Web surfing  
ence, and it offers easy-to-use apps. As an  
shines.

##### Bad:

The iPhone is something that includes a  
sting. The iPhone is something that includes a  
despite the fact that it's an integrated  
memory is content.  
ill quality for an

#### Specifications:

OS provided: Apple MacOS X; Band / mode: GSM 850/900/1800/1900

(Quadband); Wireless connectivity: IEEE 802.11b, IEEE 802.11g, IEEE 802.11n, Bluetooth v2.0 with EDR; [See full specs](#)

[See all products in the Apple iPhone series](#)

#### CNET editors' review

Reviewed by: Kent Ger

Edited by: Lindsey Turr

Reviewed on: 06/30/2008

Updated on: 07/11/2008

CNET review

Review posted on a tech blog

iP  
Pub

See my NEW iPhone 3G review

Let me start off by saying that while I'm a fan of Apple's success and products, I'm not one of those people that blindly apologizes for their products no matter what. I'll be the first to say that something works or it doesn't. My friends and many of you come to me all the time because they want my HONEST assessment. So I wanted a couple of days with the iPhone to really take it through its paces and see if this new phone is what it's hyped up to be. You must also understand that there isn't a smartphone out there that I think is perfect. As a matter of fact before the iPhone there were basically 4 smartphone OS's, Palm, Blackberry, Symbian and Windows Mobile. I stuck with Palm because it was the lesser of the 4

### Tech BLOG

- everything is plain text
- no separation between positives and negatives

# Example: iPhone



## Lab test: Apple gets iPhone 3G right for business

An abundance of new features carries iPhone 3G and iPhone 2.0 into the enterprise

By Tom Yager  
July 24, 2008

Talkback E-mail Printer Friendly Reprints Text Size A A

With the iPhone 3G's banner opening weekend and newsstands looking like a rack of brochures for the device, a review of the iPhone 3G at this point might be pro forma, except for one thing: Much of the iPhone 3G and the new iPhone 2.0 software remains an enigma to professionals and enterprises, users set apart by, among other things, their tendency to use punctuation in their e-mail. These users demand more from a handset than a cellular browser and YouTube.

**Related Stories**  
New MacBook Air: now with extra SSD goodness  
AT&T says iPhone 3G tethering coming 'soon'  
Popular Tags  
apple, iphone-3g

**[ Not everyone thinks the iPhone is enterprise-class ]**  
argues **Apple must fix 13 iPhone flaws before it's a B**

**See Also**  
iPhone delivers more misses than hits  
iPhone: The \$1,975 iPod  
**» Back to special report: Apple launches the iPhone 3G**

**The Bottom Line**  
**Apple iPhone 3G**  
Apple, [apple.com/iphone](http://apple.com/iphone)

**Very Good 8.5**  
**criteria score weight**  
Extensibility 7 20%  
Messaging 8 20%  
Networking 9 20%  
Usability 9 20%  
Multimedia 10 10%

makes it hard to imagine competitors closing the gap.  
[www.infoworld.com/article/0807020/infoworld-test/0807020\\_iphone\\_3g.html](http://www.infoworld.com/article/0807020/infoworld-test/0807020_iphone_3g.html)

**Review on InfoWorld - tech news site**

## Product summary

### The good:

The Apple iPhone has a stunning display, a sleek design, and an innovative multitouch user interface. Its Safari browser makes for a superb Web surfing experience, and it offers easy-to-use apps. As an iPod, it shines.

### The bad:

The Apple iPhone has variable call quality and lacks some basic features found in many cell phones, including stereo Bluetooth support and 3G compatibility. Integrated memory is stingy for an iPod, and you have to sync the iPhone to manage music content.

### The bottom line:

Despite some important missing features, a slow data network, and call quality that doesn't always deliver, the Apple iPhone sets a new benchmark for an integrated cell phone and MP3 player.

### Specifications:

OS provided: Apple MacOS X; Band / mode: GSM 850/900/1800/1900 (Quadband); Wireless connectivity: IEEE 802.11b, IEEE 802.11g, Bluetooth 2.0 EDR; [See full specs](#)

[See all products in the Apple iPhone series](#)

## CNET editors' review

Reviewed by: Kent Ger

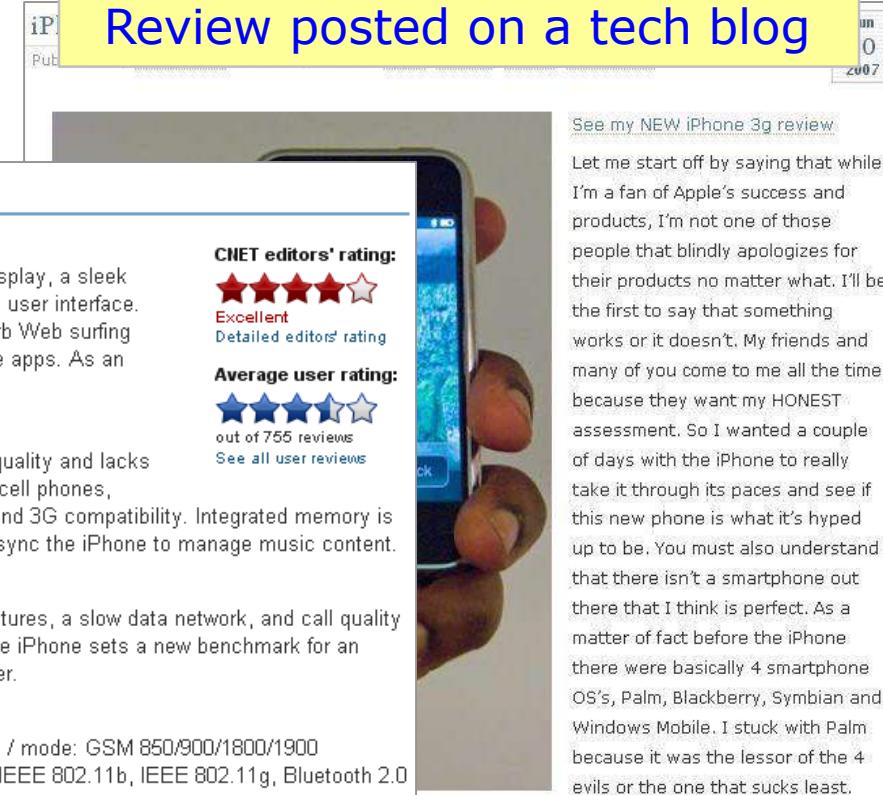
Edited by: Lindsey Turr

Reviewed on: 06/30/2008

Updated on: 07/11/2008

**CNET review**

Review posted on a tech blog



See my NEW iPhone 3g review

Let me start off by saying that while I'm a fan of Apple's success and products, I'm not one of those people that blindly apologizes for their products no matter what. I'll be the first to say that something works or it doesn't. My friends and many of you come to me all the time because they want my HONEST assessment. So I wanted a couple of days with the iPhone to really take it through its paces and see if this new phone is what it's hyped up to be. You must also understand that there isn't a smartphone out there that I think is perfect. As a matter of fact before the iPhone there were basically 4 smartphone OS's, Palm, Blackberry, Symbian and Windows Mobile. I stuck with Palm because it was the lesser of the 4 evils or the one that sucks least. Palm has a UI (user interface) that is really zero innovation. However, there are thousands of OS. Blackberry doesn't have a touch screen or tap screen or trackball or trackpad/thumb wheel. Also the Blackberry's I considered etc.) Symbian looked very promising, but I was really impressed how slow it was and that there were very few updates on him just last week right in front of me.

# Subjectivity Analysis on iPhone Reviews



## Individual's Perspective

- Highlight of what is good and bad about iPhone
    - Ex. Tech blog may contain mixture of information
  - Combination of good and bad from the different sites (*tech blog, InfoWorld and CNET*)
    - Complementing information
    - Contrasting opinions
- Ex.

CNET: *The iPhone lacks some basic features*

Tech Blog: *The iPhone has a complete set of features*

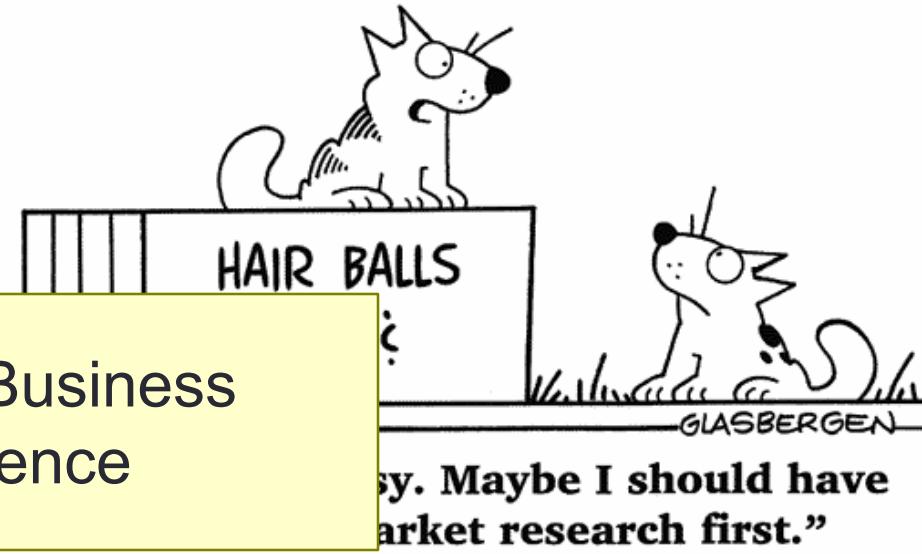
# Subjectivity Analysis on iPhone Reviews



## Business' Perspective

- Apple: What do consumers think about iPhone?
  - Do they like it?
  - What do they dislike?
  - What are the major complaints?
  - What features should we add?
- Apple's competitor:
  - What are iPhone's weaknesses?
  - How can we compete with them?
  - Do people like

Known as Business  
Intelligence



# Google Product Search



**HP Officejet 6500A Plus e-All-in-One Color Ink-jet - Fax / copier / printer / scanner**

\$89 online, \$100 nearby    377 reviews

September 2010 - Printer - HP - Inkjet - Office - Copier - Color - Scanner - Fax - 250 sh

## Reviews

**Summary** - Based on 377 reviews

1 star    2    3    4 stars    5 stars

### What people are saying

ease of use



"This was very easy to setup to four computers."

value



"Appreciate good quality at a fair price."

setup



"Overall pretty easy setup."

customer service



"I DO like honest tech support people."

size



"Pretty Paper weight."

mode



"Photos were fair on the high quality mode."

colors



"Full color prints came out with great quality."

# Bing Shopping

## HP Officejet 6500A E710N Multifunction Printer

[Product summary](#) [Find best price](#) **Customer reviews** [Specifications](#) [Related items](#)



\$121.53 - \$242.39 (14 stores)

Compare

Average rating ★★★★☆ (144)

★★★★★ (55)

★★★★☆ (54)

★★★★☆ (10)

★★★★☆ (6)

★★★★☆ (23)

★★★★☆ (0)

Most mentioned

Performance

★★★★☆ (57)

Show reviews by source

Best Buy (140)

Ease of Use

★★★★☆ (43)

CNET (5)

Print Speed

★★★★☆ (39)

Amazon.com (3)

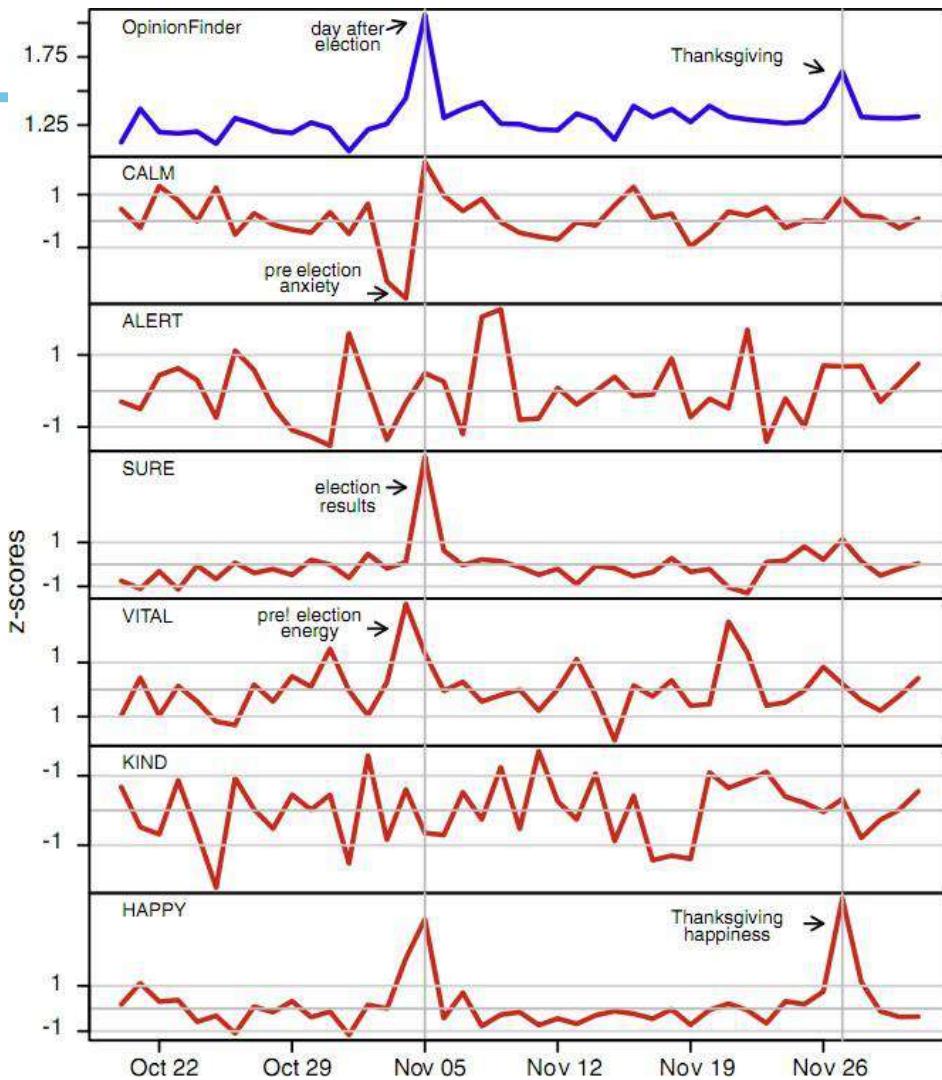
Connectivity

★★★★☆ (31)

More ▾

# Twitter sentiment:

Johan Bollen, Huina Mao, Xiaojun Zeng.  
[2011. Twitter mood predicts the stock market,](#)  
 Journal of Computational Science 2:1, 1-8. 10.1016/j.jocs.2010.12.007.



# Target Sentiment on Twitter

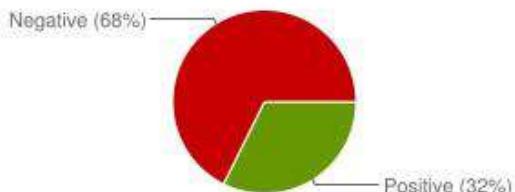
Type in a word and we'll highlight the good and the bad

- [Twitter Sentiment App](#)
- Alec Go, Richa Bhayani, Lei Huang. 2009. Twitter Sentiment Classification using Distant Supervision

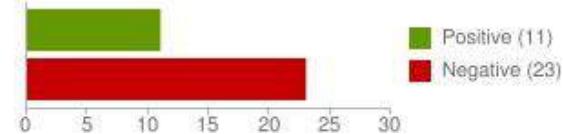
"united airlines"  [Save this search](#)

## Sentiment analysis for "united airlines"

Sentiment by Percent



Sentiment by Count



jacobson: OMG... Could @United airlines have worse customer service? W8g now 15 minutes on hold 4 questions about a flight 2DAY that need a human.  
Posted 2 hours ago

12345clumsy6789: I hate United Airlines Ceiling!!! Fukn impossible to get my conduit in this damn mess! ?  
Posted 2 hours ago

EMLandPRGbelgiu: EML/PRG fly with Q8 united airlines and 24seven to an exotic destination. <http://t.co/Z9QloAjF>  
Posted 2 hours ago

CountAdam: FANTASTIC customer service from United Airlines at XNA today. Is tweet more, but cell phones off now!  
Posted 4 hours ago

# Application Areas Summary

---

- Businesses and organizations: interested in opinions
    - product and service benchmarking
    - market intelligence
    - survey on a topic
  - Individuals: interested in other's opinions when
    - Purchasing a product
    - Using a service
    - Tracking political topics
    - Other decision making tasks
  - Ads placements: Placing ads in user-generated content
    - Place an ad when one praises a product
    - Place an ad from a competitor if one criticizes a product
  - Opinion search: providing general search for opinions
  - Text-driven forecasting: insights about other areas from text
-

# Application of sentiment analysis

- Business and organization
  - Market research



- Brand monitoring



- Customer service
- Ads placements-Social media
  - Place an ad if one praises the product
  - Place an ad from competitor if one criticizes the product
- Individual
  - Make decisions to purchase products or to use services
  - Find public opinions about political candidates and issues.

# References

---

- [https://www.mitpressjournals.org/doi/pdf/10.1162/COLI\\_a\\_00049](https://www.mitpressjournals.org/doi/pdf/10.1162/COLI_a_00049)
- <https://alphabold.com/sentiment-analysis-the-lexicon-based-approach/>
- <https://web.eecs.umich.edu/~mihalcea/papers/banea.lrec08.pdf>
- <https://www.cs.uic.edu/~liub/FBS/SentimentAnalysis-and-OpinionMining.pdf>
- [https://www.youtube.com/watch?v=OEUIzQawd1s&feature=emb\\_logo](https://www.youtube.com/watch?v=OEUIzQawd1s&feature=emb_logo)
- <https://www.kaggle.com/yommnamohamed/sentiment-analysis-using-sentiwordnet/notebook?scriptVersionId=57754564>
- <https://github.com/cjhutto/vaderSentiment/blob/master/vaderSentiment/vaderSentiment.py>

## DL for Sentiment Analysis

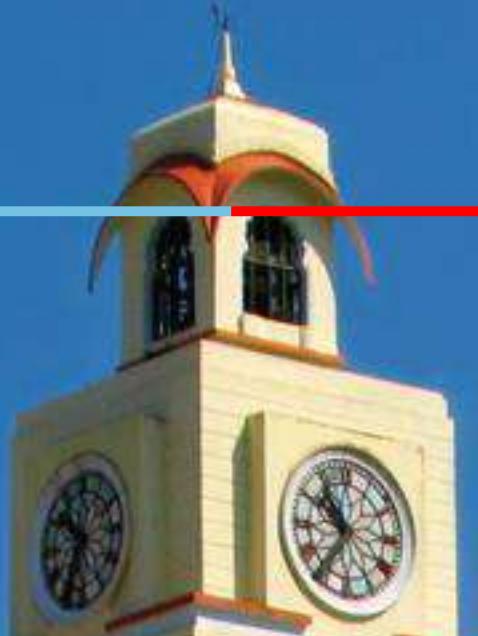
- <https://www.youtube.com/watch?v=2LiuhpKHp-k>
-



---

# Thank you





# C5: Text Mining



**BITS Pilani**  
Hyderabad Campus

Dr. Chetana Gavankar, Ph.D,  
IIT Bombay-Monash University Australia  
[Chetana.gavankar@pilani.bits-pilani.ac.in](mailto:Chetana.gavankar@pilani.bits-pilani.ac.in)



## **Session 5: Recommendation Systems**

**Date – 23<sup>rd</sup> June 2024**

**Time – 10am to 12.15pm**

These slides are prepared by the instructor, with grateful acknowledgement of Prof. Luis G. Serrano, Prof. Andrew Ng and many others who made their course materials freely available online.

# Outline

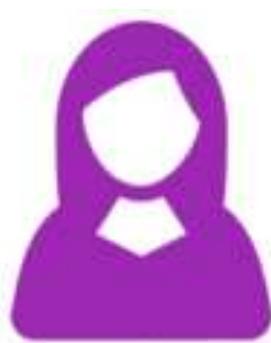
---

Recommender Systems (Ref: : <http://www.mmds.org>)

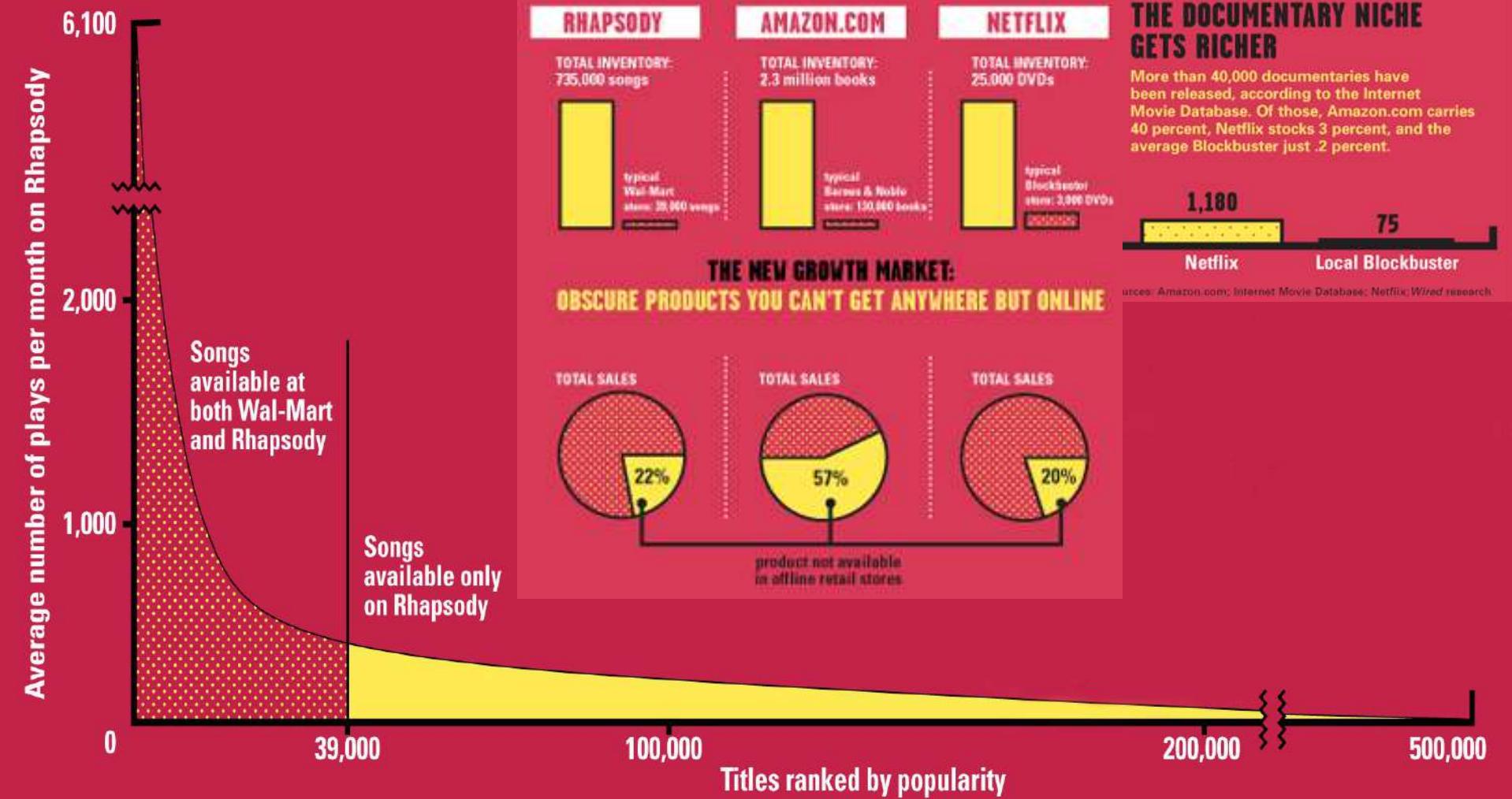
- Introduction to Recommender Systems
  - Content based Models
  - Collaborative filtering Models
    - User based Collaborative filtering
    - Item based Collaborative filtering
  - Matrix factorization using Singular Value Decomposition
  - Latent Factor Models
  - Metrics used for evaluating Recommender Systems
  - Implementing Recommender System in Python
-

# Motivation for Recommender Systems

Search Vs Recommender systems



# Sidenote: The Long Tail



Sources: Erik Brynjolfsson and Jeffrey Hu, MIT, and Michael Smith, Carnegie Mellon; Barnes & Noble; Netflix; RealNetworks  
 Source: Chris Anderson (2004)

# Commercial Interests for Recommender systems

---



- Netflix: 2/3 of the movies watched
  - Amazon: 35% sales
  - Google news: recommendations ⇒ 38% more click through
  - Choice stream: 28% of people would buy more music if they found they like the recommendation.
-

# Modelling Recommender Systems

---

- $U = \{\text{USERS}\}$
  - $I = \{\text{ITEMS}\}$
  - $F$  is a utility function, measures the usefulness of items  $I$  to user  $U$ .
  - $F: U \times I \rightarrow R$  where  $R$  is the rating
  - Characteristics of a good Utility function
    - Personalized
    - Diverse
    - Serendipity
-

# Netflix Utility Matrix



Movie/User	Movie-1	Movie-2	Movie-3	Movie-4
User-1	4	3.5	5	5
User-2	4	3	4.5	?
User-3	4.5	4	2	3

**2006-2009**

**Improvement in RMSE by 10%**

**Prize: \$1 Million**

**Netflix dataset: over 17K movies and 500K+ Users!**

# Key Problems

---

- **(1) Gathering “known” ratings for matrix**
    - How to collect the data in the utility matrix
  - **(2) Extrapolate unknown ratings from the known ones**
    - Mainly interested in high unknown ratings
      - We are not interested in knowing what you don't like but what you like
  - **(3) Evaluating extrapolation methods**
    - How to measure success/performance of recommendation methods
-

# (1) Gathering Ratings

- **Explicit**

- Ask people to rate items
  - Doesn't work well in practice – people can't be bothered

- **Implicit**

- Learn ratings from user actions
    - E.g., purchase implies high rating
  - What about low ratings?

# (2) Extrapolating Utilities

---

- **Key problem:** Utility matrix  $U$  is **sparse**
    - Most people have not rated most items
    - **Cold start:**
      - New items have no ratings
      - New users have no history
  - **Three approaches to recommender systems:**
    - 1) Content-based
    - 2) Collaborative
    - 3) Latent factor based
-

# Content-based Recommendations

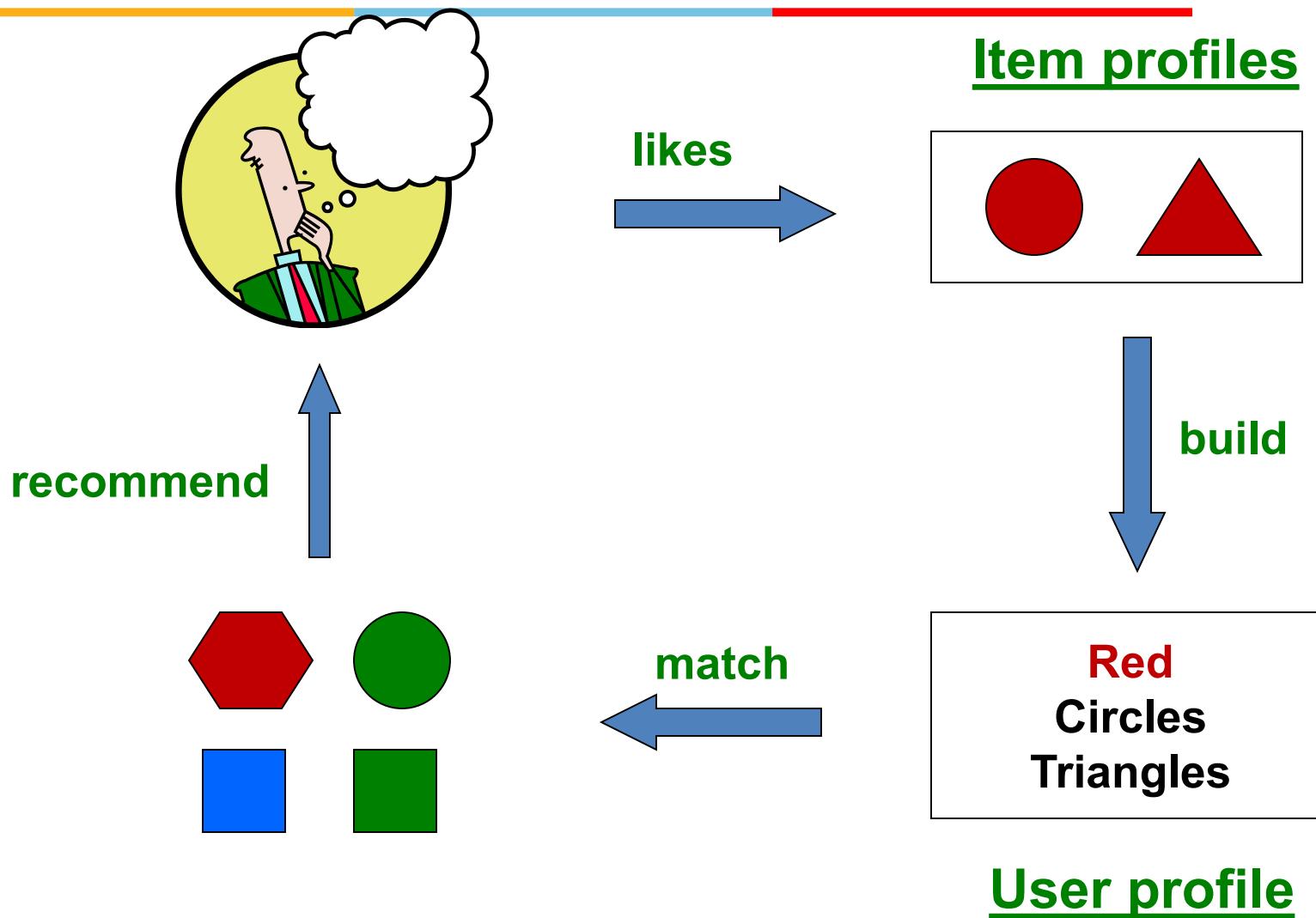
---

- **Main idea:** Recommend items to customer  $x$  similar to previous items rated highly by  $x$

*Example:*

- **Movie recommendations**
  - Recommend movies with same actor(s), director, genre, ...
- **Websites, blogs, news**
  - Recommend other sites with “similar” content

# Plan of Action



# Item Profiles

---

- For each item, create an **item profile**
- **Profile is a set (vector) of features**
  - Movies:** author, title, actor, director,...
  - Text:** Set of “important” words in document
- **How to pick important features?**
  - Usual heuristic from text mining is **TF-IDF**  
(Term frequency \* Inverse Doc Frequency)
    - Term ... Feature**
    - Document ... Item**



# Pros: Content-based Approach

---

- **+: No need for data on other users**
  - No cold-start or sparsity problems
- **+: Able to recommend to users with unique tastes**
- **+: Able to recommend new & unpopular items**
  - No first-rater problem
- **+: Able to provide explanations**
  - Can provide explanations of recommended items by listing content-features that caused an item to be recommended

# Cons: Content-based Approach

---

- -: **Finding the appropriate features is hard**
    - E.g., images, movies, music
  - -: **Recommendations for new users**
    - How to build a user profile?**
  - -: **Overspecialization**
    - Never recommends items outside user's content profile
    - People might have multiple interests
    - Unable to exploit quality judgments of other users**
-

# Baseline content based approach for rating prediction

User/Movie	Batman	Alice in Wonderland	Dumb and Dumber	Equilibrium
User A	4		3	5
User B		5	4	
User C	5	4	2	
User D	2	4		3
<b>User E</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>?</b>

Predicted Rating(E,Equilibrium) =  $\mu + b_E + b_{\text{Equilibrium}}$

Where  $\mu$  is the global average

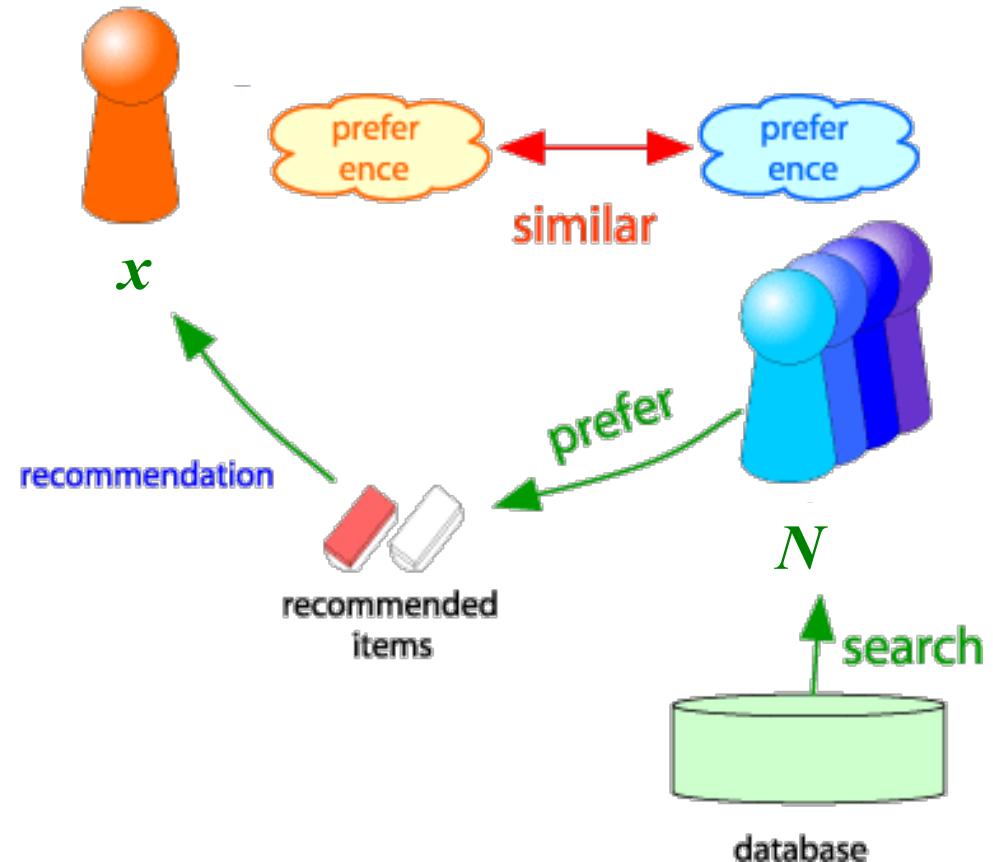
$b_E$  is the deviation of the user E

$b_{\text{Equilibrium}}$  is the deviation of the Movie Equilibrium

Predicted Rating =  $3.78+0.22+0.22 = 4.22$

# Collaborative Filtering

- Consider user  $x$
- Find set  $N$  of other users whose ratings are “similar” to  $x$ 's ratings
- Estimate  $x$ 's ratings based on ratings of users in  $N$



# Similarity Metric

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

- **Intuitively we want:**  $\text{sim}(A, B) > \text{sim}(A, C)$
- **Jaccard similarity:**  $1/5 < 2/4$
- **Cosine similarity:**  $0.386 > 0.322$

$$\cos(\mathbf{r}_x, \mathbf{r}_y) = \frac{\mathbf{r}_x \cdot \mathbf{r}_y}{\|\mathbf{r}_x\| \cdot \|\mathbf{r}_y\|}$$

- Considers missing ratings as “negative”
- **Solution: subtract the (row) mean**

# Similarity Metric

- Pearson correlation

$$Sim(x, y) = \frac{\sum_{s \in S_{xy}} (r_{xs} - \bar{r}_x)(r_{ys} - \bar{r}_y)}{\sqrt{\sum_{s \in S_{xy}} (r_{xs} - \bar{r}_x)^2} \sqrt{\sum_{s \in S_{xy}} (r_{ys} - \bar{r}_y)^2}}$$

$\bar{r}_x, \bar{r}_y \dots$  avg. rating of  $x, y$

	HP1	HP2	HP3	TW	SW1	SW2	SW3	
A	4			5	1			Mean for A= (4+5+1)/3
B	5	5	4					
C				2	4	5		
D		3					3	

	HP1	HP2	HP3	TW	SW1	SW2	SW3	
A	2/3			5/3	-7/3			Same as cosine sim. when data is centered at 0
B	1/3	1/3	-2/3					
C				-5/3	1/3	4/3		<b>sim A,B vs. A,C:</b> <b>0.092 &gt; -0.559</b>
D		0					0	

# Rating Predictions

## From similarity metric to recommendations:

- Let  $r_x$  be the vector of user  $x$ 's ratings
- Let  $N$  be the set of  $k$  users most similar to  $x$  who have rated item  $i$
- **Prediction for item  $s$  of user  $x$ :**

$$-r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi}$$

$$-r_{xi} = \frac{\sum_{y \in N} s_{xy} \cdot r_{yi}}{\sum_{y \in N} s_{xy}}$$

–Other options?

Shorthand:  
 $s_{xy} = sim(x, y)$

# Item-Item Collaborative Filtering

---

- So far: **User-user collaborative filtering**
- **Another view:** **Item-item**
  - For item  $i$ , find other similar items
  - Estimate rating for item  $i$  based on ratings for similar items
  - Can use same similarity metrics and prediction functions as in user-user model

$$r_{xi} = \frac{\sum_{j \in N(i; x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i; x)} s_{ij}}$$

$s_{ij}$ ... similarity of items  $i$  and  $j$

$r_{xj}$ ...rating of user  $u$  on item  $j$

$N(i; x)$ ... set items rated by  $x$  similar to  $i$

# Item-Item CF ( $|N|=2$ )

	users												
	1	2	3	4	5	6	7	8	9	10	11	12	$\text{sim}(1,m)$
movies	1	1		3		?	5			5		4	1.00
2				5	4			4			2	1	3
3	2	4		1	2			3		4	3	5	<u>0.41</u>
4		2	4		5				4			2	-0.10
5			4	3	4	2					2	5	-0.31
6	1			3		3			2			4	<u>0.59</u>

## Neighbor selection:

Identify movies similar to movie 1, rated by user 5

<http://www.mmds.org>

Here we use Pearson correlation as similarity:

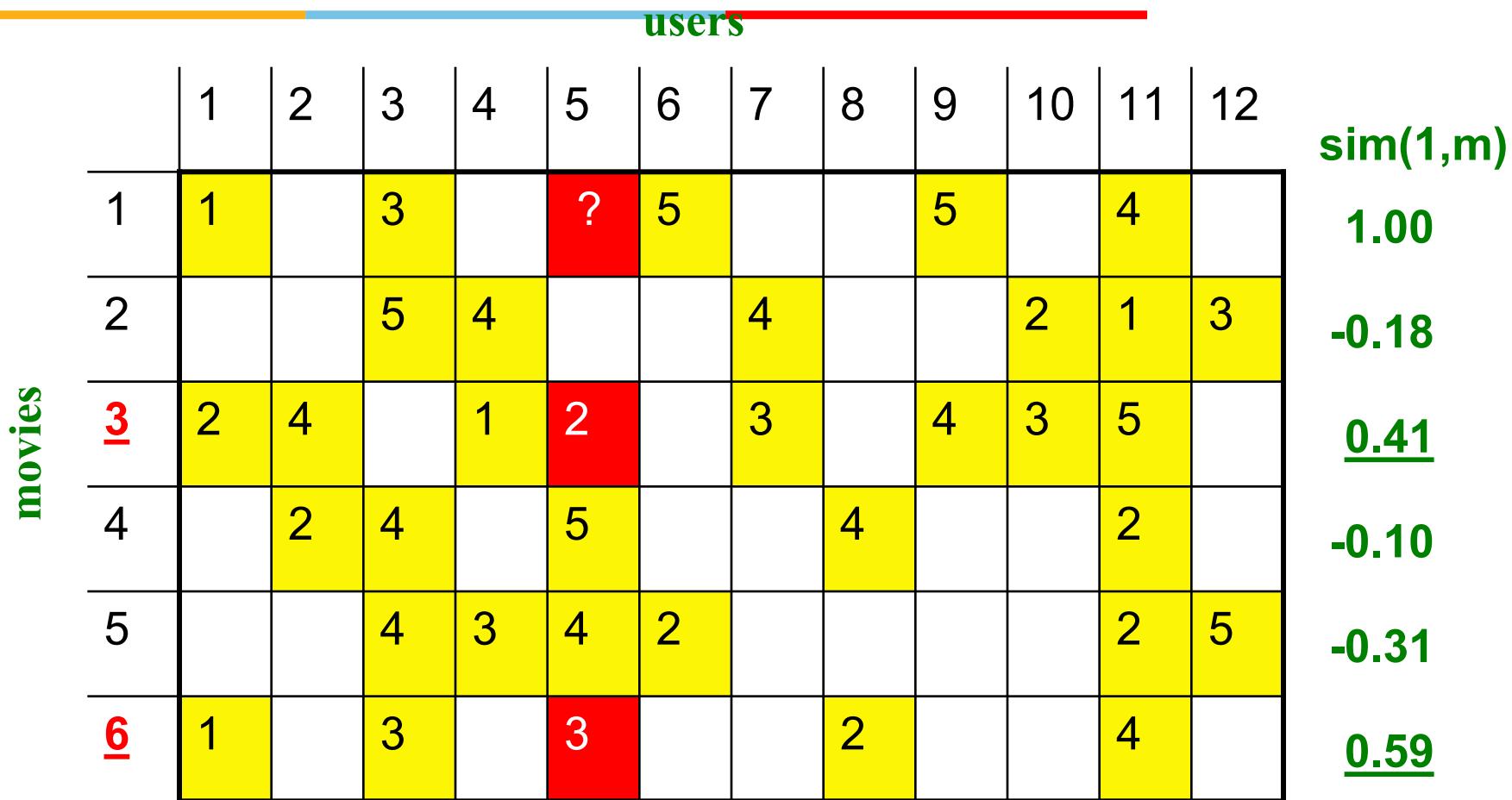
1) Subtract mean rating  $m_i$  from each movie  $i$

$$m_1 = (1+3+5+5+4)/5 = 3.6$$

row 1: [-2.6, 0, -0.6, 0, 0, 1.4, 0, 0, 1.4, 0, 0.4, 0]

2) Compute cosine similarities between rows

# Item-Item CF ( $|N|=2$ )



Compute similarity weights:

$$s_{1,3}=0.41, s_{1,6}=0.59$$

<http://www.mmds.org>

# Item-Item CF ( $|N|=2$ )

	users												
	1	2	3	4	5	6	7	8	9	10	11	12	
1	1			3		2.6	5			5		4	
2				5	4			4			2	1	3
3	2	4		1	2		3		4	3	5		
4		2	4		5			4			2		
5			4	3	4	2					2	5	
6	1			3		3			2			4	

Predict by taking weighted average:

$$r_{1,5} = (0.41*2 + 0.59*3) / (0.41+0.59) = 2.6$$

$$r_{ix} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{jx}}{\sum s_{ij}}$$

# CF: Common Practice

- Define **similarity**  $s_{ij}$  of items  $i$  and  $j$
- Select  $k$  nearest neighbors  $N(i; \mathbf{x})$ 
  - Items most similar to  $i$ , that were rated by  $\mathbf{x}$
- Estimate rating  $r_{xi}$  as the weighted average:

$$r_{xi} = b_{xi} + \frac{\sum_{j \in N(i; \mathbf{x})} s_{ij} \cdot (r_{xj} - b_{xj})}{\sum_{j \in N(i; \mathbf{x})} s_{ij}}$$

baseline estimate for  $r_{xi}$

$$b_{xi} = \mu + b_x + b_i$$

**Before:**

$$r_{xi} = \frac{\sum_{j \in N(i; \mathbf{x})} s_{ij} r_{xj}}{\sum_{j \in N(i; \mathbf{x})} s_{ij}}$$

- $\mu$  = overall mean movie rating
- $b_x$  = rating deviation of user  $\mathbf{x}$   
 $= (\text{avg. rating of user } \mathbf{x}) - \mu$
- $b_i$  = rating deviation of movie  $i$

# Item-Item vs. User-User

	Avatar	LOTR	Matrix	Pirates
Alice	1		0.8	
Bob		0.5		0.3
Carol	0.9		1	0.8
David			1	0.4

- In practice, it has been observed that item-item often works better than user-user
- Why? Items are simpler, users have multiple tastes

# Pros/Cons of Collaborative Filtering

- **+ Works for any kind of item**
  - No feature selection needed
- **- Cold Start:**
  - Need enough users in the system to find a match
- **- Sparsity:**
  - The user/ratings matrix is sparse
  - Hard to find users that have rated the same items
- **- First rater:**
  - Cannot recommend an item that has not been previously rated
  - New items, Esoteric items
- **- Popularity bias:**
  - Cannot recommend items to someone with unique taste
  - Tends to recommend popular items

# The Netflix Prize

- **Training data**

- 100 million ratings, 480,000 users, 17,770 movies
  - 6 years of data: 2000-2005

- **Test data**

- Last few ratings of each user (2.8 million)
  - **Evaluation criterion:** Root Mean Square Error (RMSE) =

$$\frac{1}{|R|} \sqrt{\sum_{(i,x) \in R} (\hat{r}_{xi} - r_{xi})^2}$$

- Netflix's system RMSE: **0.9514**

- **Competition**

- 2,700+ teams
  - **\$1 million** prize for 10% improvement on Netflix

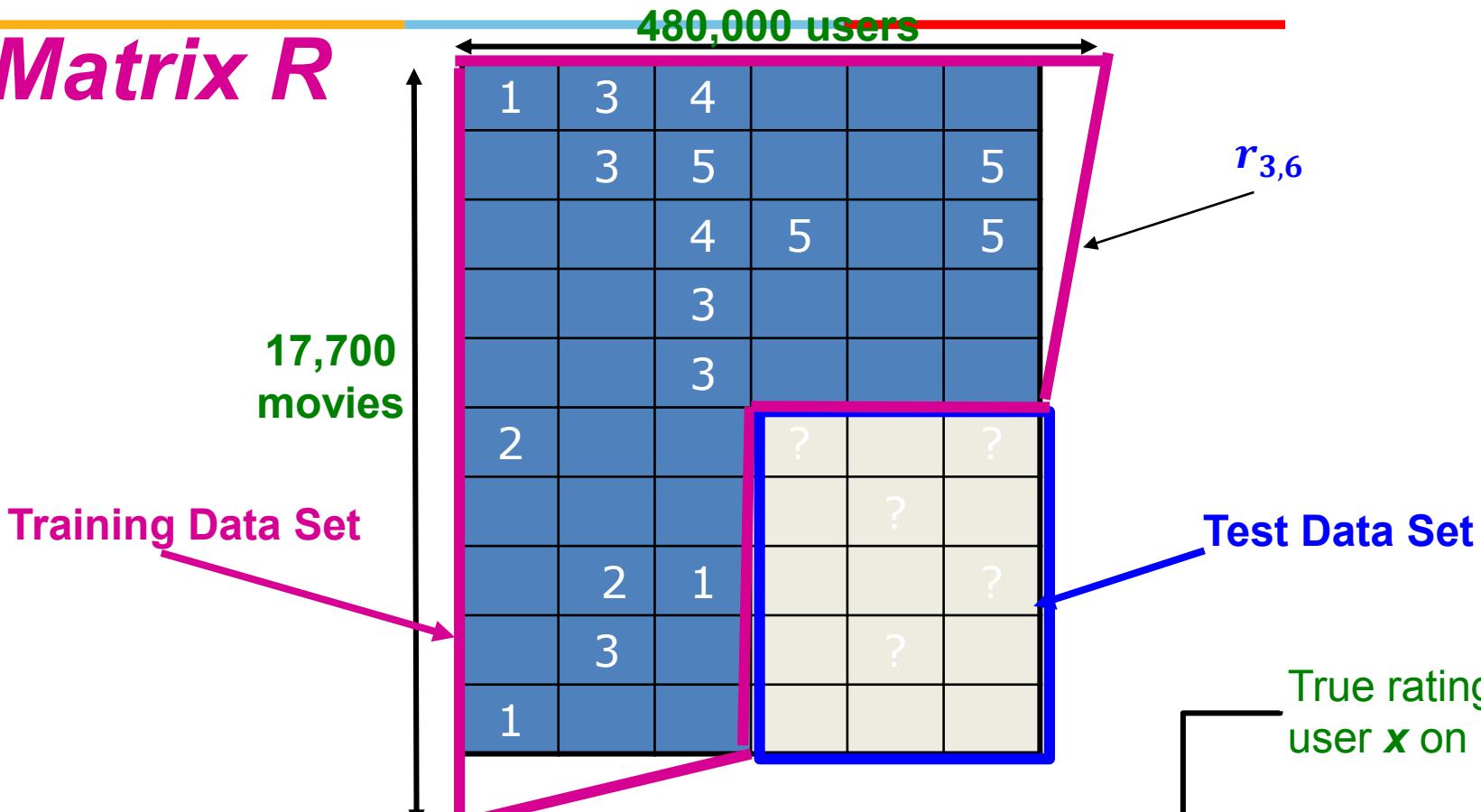
# The Netflix Utility Matrix $R$

**Matrix  $R$**

480,000 users					
17,700 movies	1	3	4		
		3	5		5
			4	5	5
				3	
				3	
	2			2	2
					5
		2	1		1
				3	
	1				

# Utility Matrix $R$ : Evaluation

**Matrix  $R$**



$$\text{RMSE} = \frac{1}{|R|} \sqrt{\sum_{(i,x) \in R} (\hat{r}_{xi} - r_{xi})^2}$$

Predicted rating

# Modeling Local & Global Effects

- **Global:**

- Mean movie rating: **3.7 stars**
- *The Sixth Sense* is **0.5** stars above avg.
- Joe rates **0.2** stars below avg.

⇒ **Baseline estimation:**

**Joe will rate *The Sixth Sense* 4 stars**

- **Local neighborhood (CF/NN):**

- Joe didn't like related movie *Signs*

⇒ **Final estimate:**

**Joe will rate *The Sixth Sense* 3.8 stars**



# Modeling Local & Global Effects

- In practice we get better estimates if we model deviations:

$$\hat{r}_{xi} = b_{xi} + \frac{\sum_{j \in N(i,x)} s_{ij} \cdot (r_{xj} - b_{xj})}{\sum_{j \in N(i,x)} s_{ij}}$$

baseline estimate for  $r_{xi}$

$$b_{xi} = \mu + b_x + b_i$$

$\mu$  = overall mean rating

$b_x$  = rating deviation of user  $x$   
 $= (\text{avg. rating of user } x) - \mu$

$b_i$  = (avg. rating of movie  $i$ ) –  $\mu$

## Problems/Issues:

- 1) Similarity measures are “arbitrary”
- 2) Pairwise similarities neglect interdependencies among users
- 3) Taking a weighted average can be restricting

**Solution:** Instead of  $s_{ij}$  use  $w_{ij}$  that we estimate directly from data

# Idea: Interpolation Weights $w_{ij}$

---

- Use a **weighted sum** rather than **weighted avg.**:

$$\widehat{r}_{xi} = b_{xi} + \sum_{j \in N(i;x)} w_{ij}(r_{xj} - b_{xj})$$

- **A few notes:**

- $N(i; x)$  ... set of movies rated by user  $x$  that are similar to movie  $i$

- $w_{ij}$  is the interpolation weight (some real number)

  - We allow:  $\sum_{j \in N(i,x)} w_{ij} \neq 1$

- $w_{ij}$  models interaction between pairs of movies  
(it does not depend on user  $x$ )

# Idea: Interpolation Weights $w_{ij}$

---

- $\hat{r}_{xi} = b_{xi} + \sum_{j \in N(i,x)} w_{ij} (r_{xj} - b_{xj})$
- How to set  $w_{ij}$ ?

– Remember, error metric is:  $\frac{1}{|R|} \sqrt{\sum_{(i,x) \in R} (\hat{r}_{xi} - r_{xi})^2}$  or equivalently SSE:  $\sum_{(i,x) \in R} (\hat{r}_{xi} - r_{xi})^2$

– Find  $w_{ij}$  that minimize SSE on training data!

- Models relationships between item  $i$  and its neighbors  $j$
- $w_{ij}$  can be learned/estimated based on  $x$  and all other users that rated  $i$

# Recommendations via Optimization

---

- **Goal:** Make good recommendations
  - Quantify goodness using **RMSE**:  
**Lower RMSE  $\Rightarrow$  better recommendations**
  - Want to make good recommendations on items that user has not yet seen. **Can't really do this!**
  - **Let's set build a system such that it works well on known (user, item) ratings**  
And **hope** the system will also predict well the **unknown ratings**

1	3	4		
3	5			5
	4	5		5
	3			
	3			
2		2	2	
			5	
	2	1		1
	3		3	
1				

# Recommendations via Optimization

- Idea: Let's set values  $w$  such that they work well on known (user, item) ratings
- How to find such values  $w$ ?
- Idea: Define an objective function and solve the optimization problem
- Find  $w_{ij}$  that minimize SSE on training data!

$$J(w) = \sum_{x,i} \left( \left[ b_{xi} + \sum_{j \in N(i;x)} w_{ij}(r_{xj} - b_{xj}) \right] - r_{xi} \right)^2$$

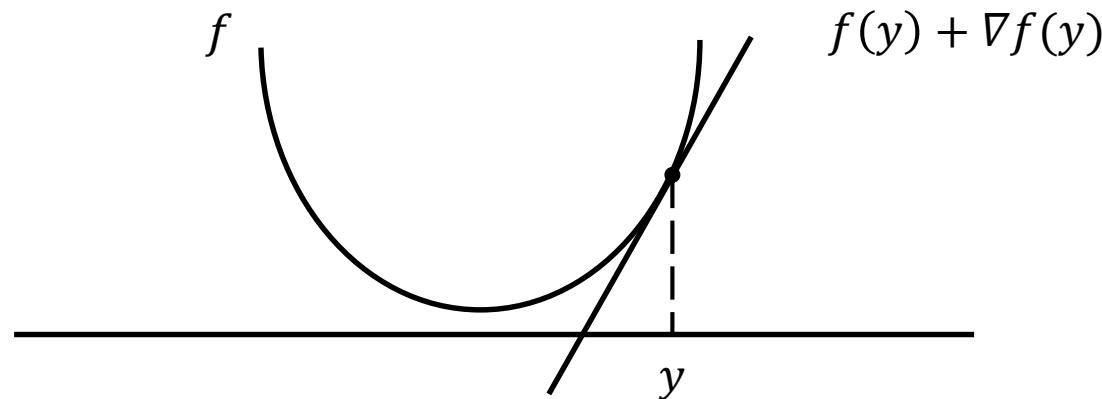
Predicted rating
  

True rating

- Think of  $w$  as a vector of numbers

# Detour: Minimizing a function

- **A simple way to minimize a function  $f(x)$ :**
  - Compute the take a derivative  $\nabla f$
  - Start at some point  $y$  and evaluate  $\nabla f(y)$
  - Make a step in the reverse direction of the gradient:
$$y = y - \nabla f(y)$$
- Repeat until converged



# Interpolation Weights

- **Gradient decent:**

$$J(w) = \sum_x \left( \left[ b_{xi} + \sum_{j \in N(i;x)} w_{ij} (r_{xj} - b_{xj}) \right] - r_{xi} \right)^2$$

–Iterate until convergence:  $w \leftarrow w - \eta \nabla_w J$

–where  $\nabla_w J$  is the gradient (derivative evaluated on data):

$$\nabla_w J = \left[ \frac{\partial J(w)}{\partial w_{ij}} \right] = 2 \sum_{x,i} \left( \left[ b_{xi} + \sum_{k \in N(i;x)} w_{ik} (r_{xk} - b_{xk}) \right] - r_{xi} \right) (r_{xj} - b_{xj})$$

for  $j \in \{N(i; x), \forall i, \forall x\}$

else  $\frac{\partial J(w)}{\partial w_{ij}} = 0$

–Note: We fix movie  $i$ , go over all  $r_{xi}$ , for every movie  $j \in N(i; x)$ , we compute  $\frac{\partial J(w)}{\partial w_{ij}}$

**while**  $|w_{new} - w_{old}| > \varepsilon$ :

$$w_{old} = w_{new}$$

$$w_{new} = w_{old} - \eta \cdot \nabla_w w_{old}$$

$\eta$  ... learning rate

# SVD Theorem

---

$$\mathbf{A}_{[m \times n]} = \mathbf{U}_{[m \times r]} \Sigma_{[r \times r]} (\mathbf{V}_{[n \times r]})^T$$

**A: Input data matrix**

- $m \times n$  matrix (e.g.,  $m$  users,  $n$  movies)

**U: Left singular vectors**

- $m \times r$  matrix ( $m$  users,  $r$  concepts)

**$\Sigma$ : Singular values**

- $r \times r$  diagonal matrix (strength of each ‘concept’)  
( $r$  : rank of the matrix  $\mathbf{A}$ )

**V: Right singular vectors**

- $n \times r$  matrix ( $n$  movies,  $r$  concepts)
-

# Singular Value Decomposition

---

- The key issue in an SVD decomposition is to **find a lower dimensional feature space** where the **new features represent “concepts”** and the **strength of each concept** in the context of the collection is computable.
  - The core of the SVD algorithm lies in the following theorem
  - It is always possible to decompose a given matrix  $A$  into  $A = U \Sigma V^T$ .
-

# Example

$$\begin{bmatrix} 4 & 1 & 1 & 4 \\ 1 & 4 & 2 & 0 \\ 2 & 1 & 4 & 5 \end{bmatrix} = \begin{bmatrix} -0.61 & 0.28 & -0.74 \\ -0.29 & -0.95 & -0.12 \\ -0.74 & 0.14 & .66 \end{bmatrix} \times \begin{bmatrix} 8.87 & 0.00 & 0.00 & 0.00 \\ 0.00 & 4.01 & 0.00 & 0.00 \\ 0.00 & 0.00 & 2.51 & 0.00 \end{bmatrix} \times \begin{bmatrix} -0.47 & -0.28 & -0.47 & -0.69 \\ 0.11 & -0.85 & -0.27 & 0.45 \\ -0.71 & -0.23 & 0.66 & 0.13 \\ -0.52 & 0.39 & -0.53 & 0.55 \end{bmatrix}$$

User to Movie  
Utility Matrix

User to Concept

Strength of each concept

Movie to Concept

$$\begin{bmatrix} 2.69 & 0.57 & 2.22 & 4.25 \\ 0.78 & 3.93 & 2.21 & 0.04 \\ 3.17 & 1.38 & 2.92 & 4.78 \end{bmatrix} = \begin{bmatrix} -0.61 & 0.28 \\ -0.29 & -0.95 \\ -0.74 & 0.14 \end{bmatrix} \times \begin{bmatrix} 8.87 & 0.00 \\ 0.00 & 4.01 \end{bmatrix} \times \begin{bmatrix} -0.47 & -0.28 & -0.47 & -0.69 \\ 0.11 & -0.85 & -0.27 & 0.45 \end{bmatrix}$$

Reconstructed Matrix

User to Concept

Strength of each concept

Movie to Concept

# Recommendations for new User

- How to use SVD for recommendations?

$$\mathbf{u}_{new} = \mathbf{u} \times V_{m \times r} \times S^{-1}_{r \times r}$$

$$\begin{bmatrix} 8.87 & 0.00 \\ 0.00 & 4.01 \\ 0.00 & 0.00 \end{bmatrix}$$

Strength of each concept

$$\begin{bmatrix} -0.23 & -0.89 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} -0.47 & 0.11 \\ -0.28 & -0.85 \\ -0.47 & -0.27 \\ -0.69 & 0.45 \end{bmatrix} \times \begin{bmatrix} 0.11 & 0.00 \\ 0.00 & 0.25 \end{bmatrix}$$

$$\begin{bmatrix} -0.47 & -0.28 & -0.47 & -0.69 \\ 0.11 & -0.85 & -0.27 & 0.45 \end{bmatrix}$$

Movie to Concept

# Drawbacks of SVD

---

- Conventional SVD is undefined for incomplete matrices!
  - Imputation to fill in missing values
  - Increases the amount of data
  - We need an approach that can simply ignore missing ratings
-

# Drawback of SVD

---

- Though SVD gives us the best rank approximation, the major problem is that it is **not defined for missing entries**.
  - Hence the Latent Factor Model comes handy which draws inspiration from SVD.
-

# User – Movie interactions in real world

User/ Movie	M1	M2	M3	M4	M5
User1	1	3	2	5	4
User2	2	1	1	1	5
User3	3	2	3	1	5
User4	2	4	1	5	2

User/ Movie	M1	M2	M3	M4	M5
User1	4	4	4	4	4
User2	4	4	4	4	4
User3	4	4	4	4	4
User4	4	4	4	4	4

User/ Movie	M1	M2	M3	M4	M5
User1	1	3	2	5	4
User2	2	1	1	1	5
User3	3	2	3	1	5
User4	2	4	1	5	2

User/ Movie	M1	M2	M3	M4	M5
User1	3	1	1	3	1
User2	1	2	4	1	3
User3	3	1	1	3	1
User4	4	3	5	4	4

# Movie / User Features

## ➤ Movie Features



**Comedy**



**Action**



**Horror**



**Drama**

## ➤ User Preferences

- Likeliness of genre of movies (0/1)

# Movie / User Features

Movie/User	Comedy	Action
M1	3	1
M2	1	2
M3	1	4
M4	3	1
M5	1	3

User/Movie	Comedy	Action
User1	1	0
User2	0	1
User3	1	0
User4	1	1

# Latent Factors using Matrix Factorization

Movie/ Features	M1	M2	M3	M4	M5
Comedy	3	1	1	3	1
Action	1	2	4	1	3

User/Features	Comedy	Action
User1	1	0
User2	0	1
User3	1	0
User4	1	1

User/Movie	M1	M2	M3	M4	M5
User1	3	1	1	3	1
User2	1	2	4	1	3
User3	3	1	1	3	1
User4	4	3	5	4	4

# Stochastic Gradient Descent (SGD)

Movie/ Features	M1	M2	M3	M 4	M 5
Comedy	1.2	3.1	0.3	2.5	0.2
Action	2.4	1.5	4.4	0.4	1.1

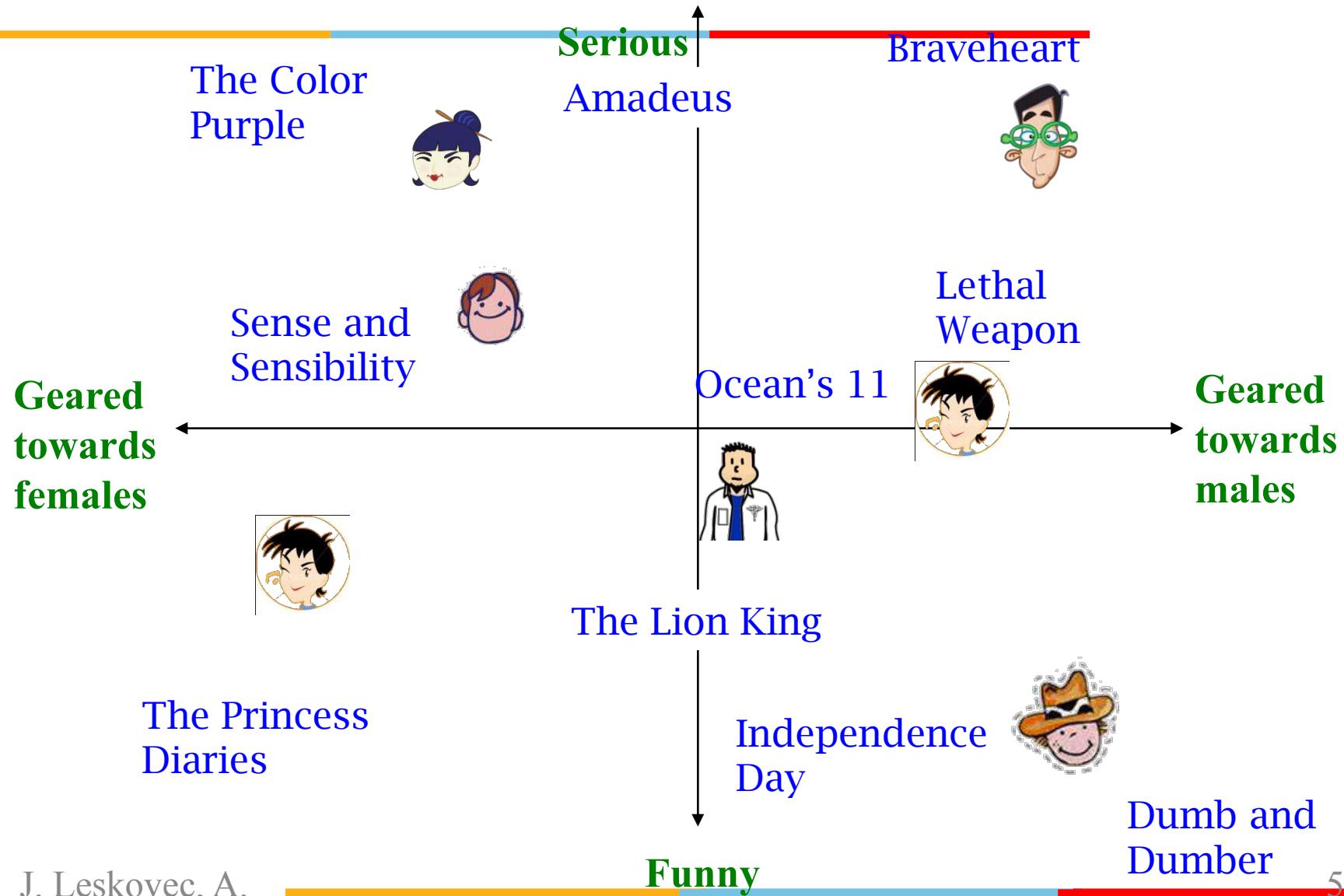
User/Fe atures	F1	F2
User1	0.2	0.5
User2	0.3	0.4
User3	0.7	0.8
User4	0.4	0.5

User/Movie	M1	M2	M3	M4	M5
User1	1.44	1.37	2.26	0.7	0.59
User2	1.32	1.53	1.85	0.91	0.5
User3	2.76	3.37	3.73	2.07	1.02
User4	1.68	1.99	2.32	1.2	0.63



User/ Movie	M1	M2	M3	M4	M5
User1	3	1	1	3	1
User2	1	2	4	1	3
User3	3	1	1	3	1
User4	4	3	5	4	4

# Latent Factor Models (e.g., SVD)



# Latent Factor Models

- “SVD” on Netflix data:  $\mathbf{R} \approx \mathbf{Q} \cdot \mathbf{P}^T$

$$\text{SVD: } A = U \Sigma V^T$$

		users				factors					
		items				swimmers					
1		3		5		5		4			
		5	4		4		2	1	3		
2	4		1	2		3	4	3	5		
	2	4		5		4		2			
	4	3	4	2				2	5		
1		3	3		2			4			

$\approx$

		users				factors					
		.1	-.4	.2							
		-.5	.6	.5							
		-.2	.3	.5							
		1.1	2.1	.3							
		-.7	2.1	-2							
		-1	.7	.3							

$\mathbf{R}$        $\mathbf{Q}$

		users				factors					
1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	

$\mathbf{P}^T$

- For now let's assume we can approximate the rating matrix  $\mathbf{R}$  as a product of “thin”  $\mathbf{Q} \cdot \mathbf{P}^T$

– $\mathbf{R}$  has missing entries but let's ignore that for now!

- Basically, we will want the reconstruction error to be small on known ratings and we don't care about the values on the missing ones

# Ratings as Products of Factors

- How to estimate the missing rating of user  $x$  for item  $i$ ?

		users									
		1	3			5		5		4	
items	1		5	4	?	4		2	1	3	
	2	4		1	2	3	4	3	5		
	2	4		5		4			2		
	4	3	4	2					2	5	
	1	3	3			2			4		

≈

$$\hat{r}_{xi} = q_i \cdot p_x$$

$$= \sum_f q_{if} \cdot p_{xf}$$

$q_i$  = row  $i$  of  $Q$   
 $p_x$  = column  $x$  of  $P^T$

		.1	-.4	.2
		-.5	.6	.5
items	items	-.2	.3	.5
	1.1	2.1	.3	
	-.7	2.1	-2	
	-1	.7	.3	

• factors

$Q$

		users											
		1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
factors	1	-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
	2	2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1
	3												

$P^T$

# Ratings as Products of Factors

- How to estimate the missing rating of user  $x$  for item  $i$ ?

users

1		3			5			5		4	
		5	4	?	4			2	1	3	
2	4		1	2		3		4	3	5	
	2	4		5			4			2	
		4	3	4	2					2	5
1		3		3			2			4	

≈

$$\hat{r}_{xi} = q_i \cdot p_x$$

$$= \sum_f q_{if} \cdot p_{xf}$$

$q_i$  = row  $i$  of  $Q$   
 $p_x$  = column  $x$  of  $P^T$

items

.1	-.4	.2
<b>-.5</b>	<b>.6</b>	<b>.5</b>
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

• factors

users

1.1	-.2	.3	.5	<b>-.2</b>	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	<b>.3</b>	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	<b>2.4</b>	.9	-.3	.4	.8	.7	-.6	.1

$P^T$

# Ratings as Products of Factors

- How to estimate the missing rating of user  $x$  for item  $i$ ?

users

1		3		5		5		4	
		5	4	2.4	4		2	1	3
2	4		1	2	3	4	3	5	
	2	4		5		4		2	
		4	3	4	2			2	5
1		3		3		2		4	

≈

$$\hat{r}_{xi} = q_i \cdot p_x^T$$

$$= \sum_f q_{if} \cdot p_{xf}$$

$q_i$  = row  $i$  of  $Q$   
 $p_x$  = column  $x$  of  $P^T$

items

.1	-.4	.2
-5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

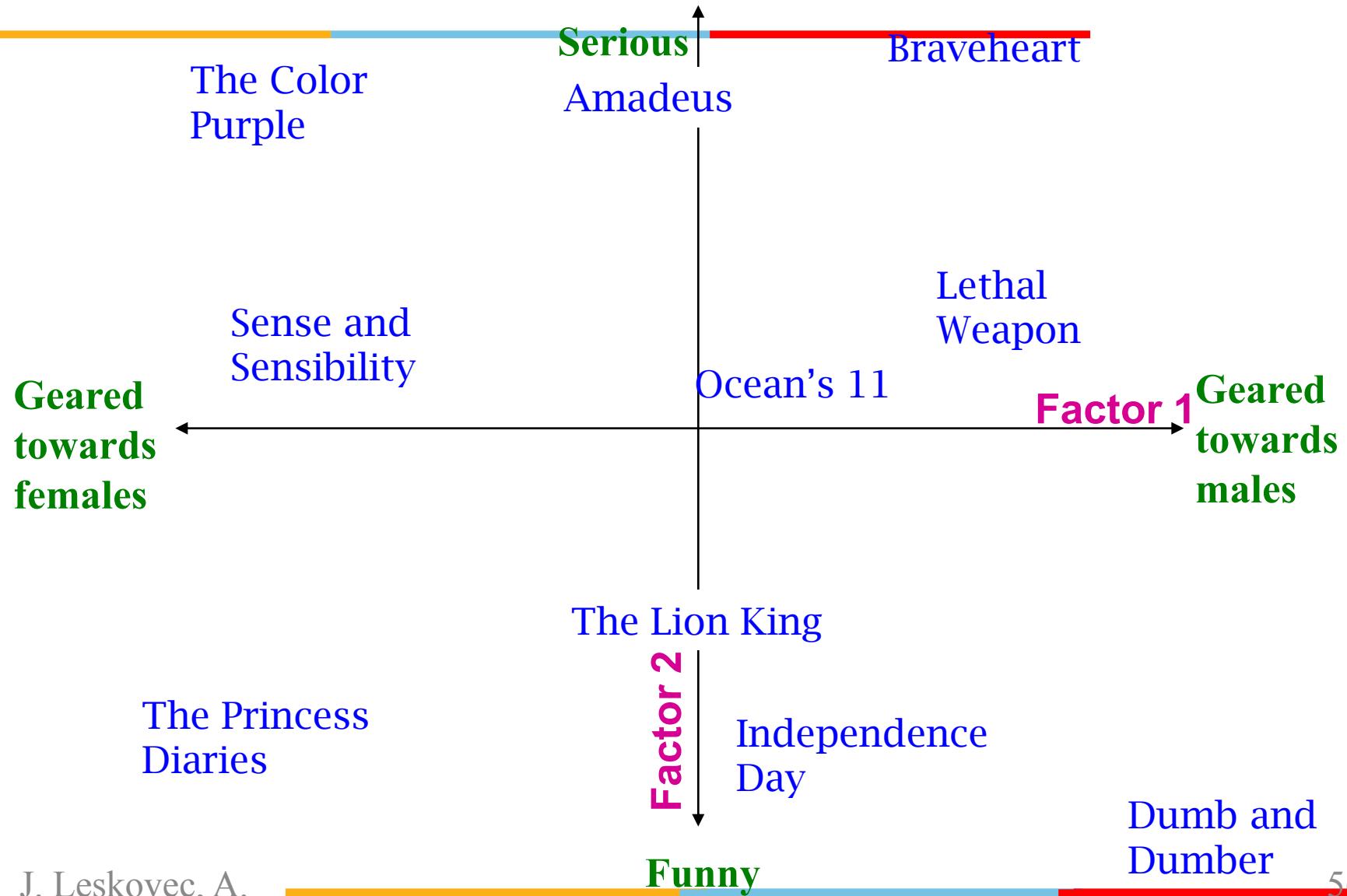
•  $f$  factors

users

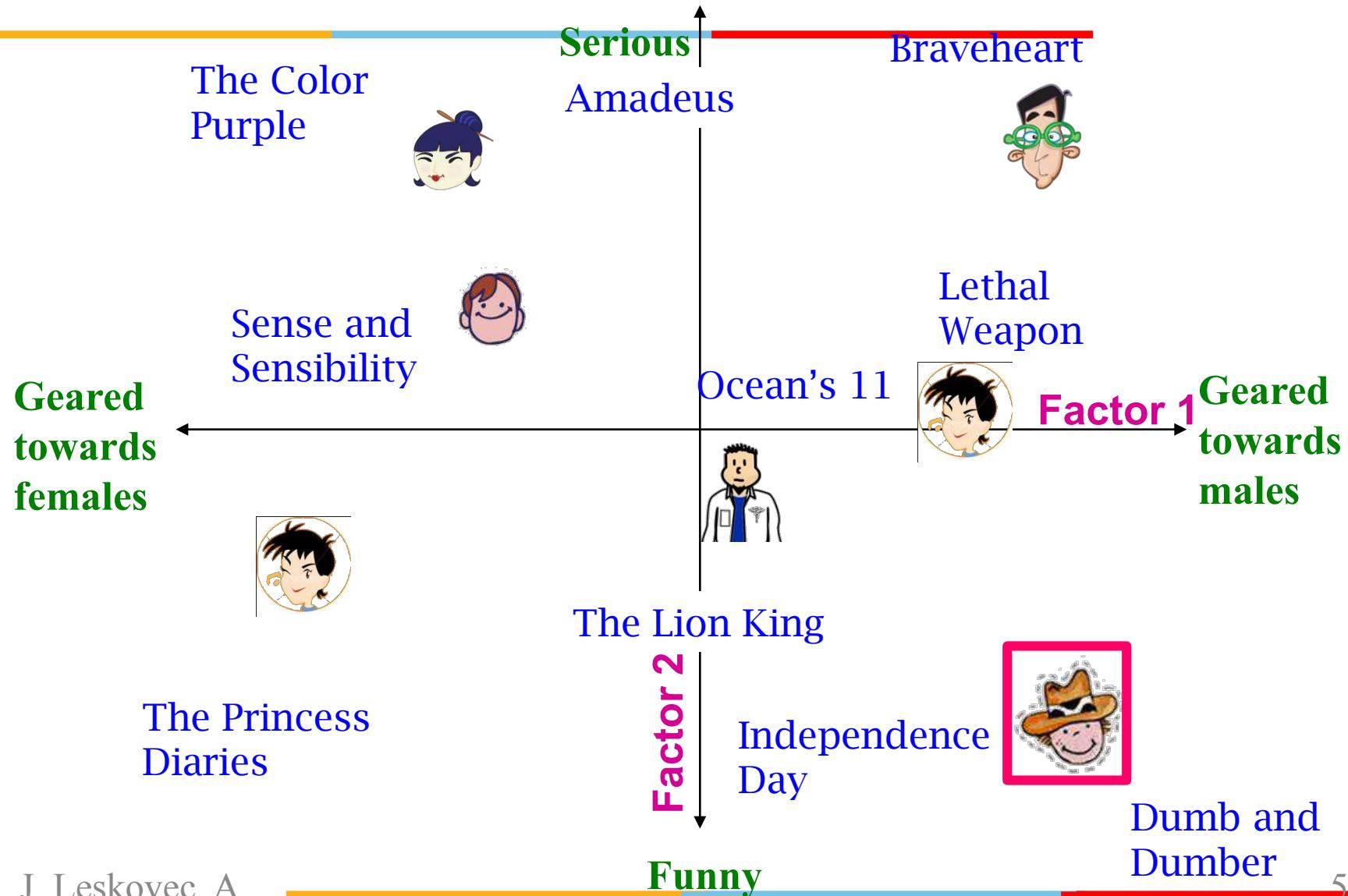
1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1

$P^T$

# Latent Factor Models



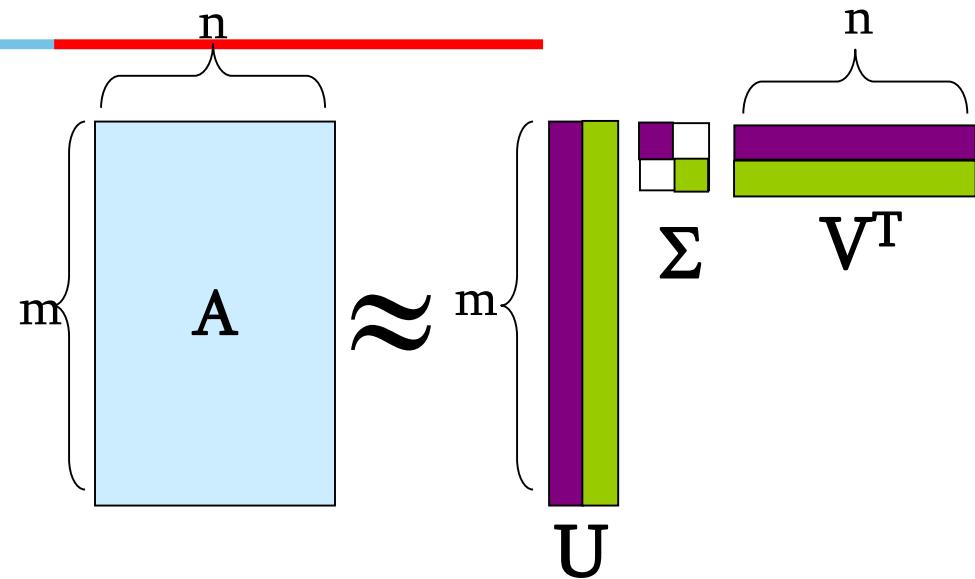
# Latent Factor Models



# Recap: SVD

- **Remember SVD:**

- $A$ : Input data matrix
- $U$ : Left singular vecs
- $V$ : Right singular vecs
- $\Sigma$ : Singular values



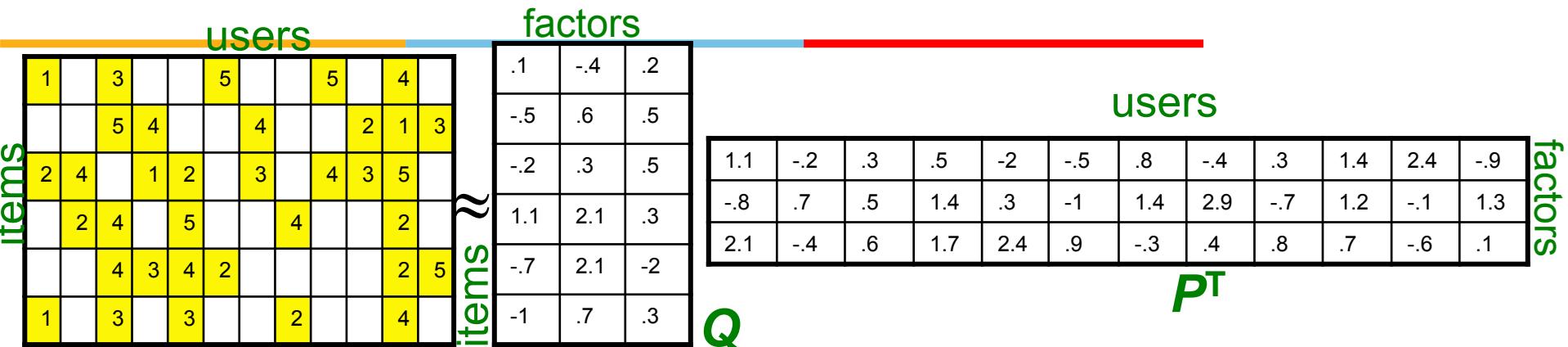
- **So in our case:**

“SVD” on Netflix data:  $R \approx Q \cdot P^T$

$$A = R, \quad Q = U, \quad P^T = \Sigma V^T$$

$$\hat{r}_{xi} = q_i \cdot p_x$$

# Latent Factor Models



- SVD isn't defined when entries are missing!
  - Use specialized methods to find  $P$ ,  $Q$

$$-\min_{P,Q} \sum_{(i,x) \in R} (r_{xi} - q_i \cdot p_x^T)^2$$

– Note:

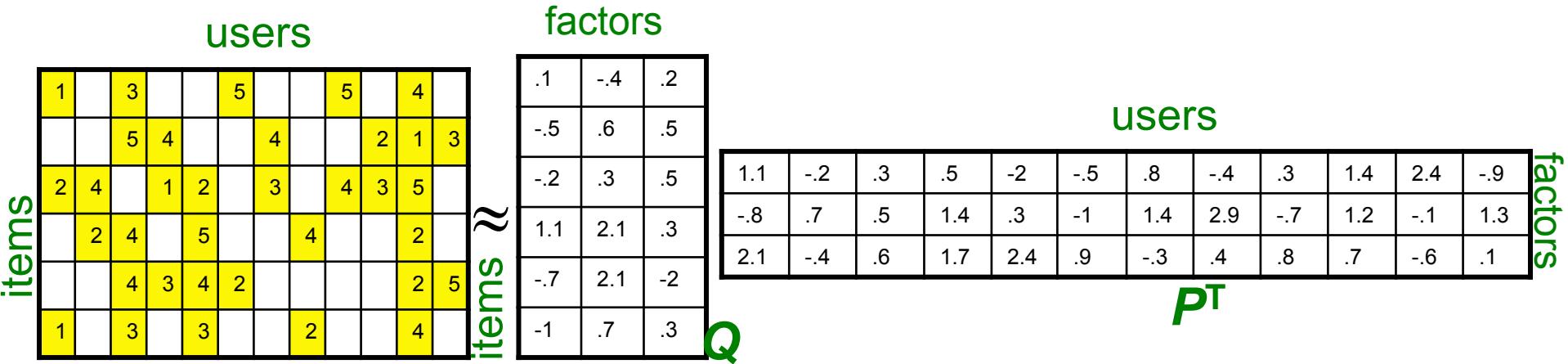
- We don't require cols of  $P$ ,  $Q$  to be orthogonal/unit length
  - $P$ ,  $Q$  map users/movies to a latent space
  - The most popular model among Netflix contestants

$$\hat{r}_{xi} = q_i \cdot p_x^T$$

# Latent Factor Models

- Our goal is to find  $P$  and  $Q$  such that:

$$\min_{P,Q} \sum_{(i,x) \in R} (r_{xi} - q_i \cdot p_x^T)^2$$



# Back to Our Problem

- Want to minimize SSE for unseen test data
- Idea: Minimize SSE on training data
  - Want large  $k$  (# of factors) to capture all the signals
  - But, SSE on test data begins to rise for  $k > 2$
- This is a classical example of **overfitting**:
  - With too much freedom (too many free parameters) the model starts fitting noise
    - That is it fits too well the training data and thus **not generalizing** well to unseen test data

1	3	4				
	3	5				5
		4	5			5
			3			
			3			
2				?		?
	2	1			?	?
	3				?	
1						

# Dealing with Missing Entries

- **To solve overfitting we introduce regularization:**
    - Allow rich model where there are sufficient data
    - Shrink aggressively where data are scarce

1	3	4		
	3	5		5
	4	5		5
	3			
	3			
2			?	?
			?	?
	2	1		?
	3		?	
1				

$$\min_{P,Q} \underbrace{\sum_{\text{training}} (r_{xi} - q_i p_x)^2}_{\text{"error"}} + \left[ \lambda_1 \underbrace{\sum_x \|p_x\|^2}_x + \lambda_2 \underbrace{\sum_i \|q_i\|^2}_i \right]$$

$\lambda_1, \lambda_2, \dots$  user set regularization parameters

**Note:** We do not care about the “raw” value of the objective function, but we care in P,Q that achieve the minimum of the objective

# Rating predictions

User/Features	Comedy	Action
User1	1	0
User2	0	1
User3	1	0
User4	1	1

Movie/Features	M1	M2	M3	M4	M5
Comedy	3	1	1	3	1
Action	1	2	4	1	3

User/Movie	M1	M2	M3	M4	M5
User1	3		1		1
User2	1		4	1	
User3	3	1		3	1
User4		3		4	4

# Objective function

Our goal is to find two matrices P and Q such that the following objective function is minimized on **test data**:

User/Features	Comedy	Action
User1	1	0
User2	0	1
User3	1	0
User4	1	1

Movie/ Features	M1	M2	M3	M4	M5
Comedy	3	1	1	3	1
Action	1	2	4	1	3

User/Movie	M1	M2	M3	M4	M5
User1	3		1		1
User2	1		4	1	
User3	3	1		3	1
User4		3		4	4

# Accuracy of estimated rating / Error Based

- Mean Absolute Error (MAE)

$$MAE = \frac{\sum_{i=1}^N |p_i - r_i|}{N}$$

- Mean square error (MSE)

- Root Mean Squared Error(RMSE)

User/Movie	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6
User1	2			4	4	
User2	5		4			1
User3			5		2	
User4		1		5		4
User5			4			2
User6	4	5		1		

Training Set

Test Set

# Accuracy of estimated rating / Error Based



User Id	Movie Id	Actual	Predicted	MAE	MSE	RMSE
User1	4	4	2	2	4	4
User1	5	4	1	3	9	9
User2	6	1	1.5	0.5	0.25	0.25
User3	5	2	2	0	0	0
User4	4	5	4.5	0.5	0.25	0.25
User4	6	4	3	1	1	1
User5	6	2	2	0	0	0
User6	4	1	3	2	4	4
				9/8	18.5/8	Sqrt(18.5/8)

# Precision at rank K

Movie	User 1	Actual	Predicted
1	1	4	2.3
2	1	2	3.6
3	1	3	3.4
4	1	?	4.4
5	1	5	4.5
6	1	?	2.3
7	1	2	4.9
8	1	?	4.3
9	1	?	3.3
10	1	4	4.3

Ignore the values whose actual is not known and sort the items in descending order of the predicted rating.

Item7 2/4.9

item5 5/4.5

item10 4/4.3

item2 2/3.6

item3 3/3.4

item1 4/2.3

Let's assume that all rating  $\geq 3.5$  are relevant then the recommender system will suggest the following

Item7 2/4.9

item5 5/4.5

item10 4/4.3

# Problems with Error Measures

---

- **Narrow focus on accuracy sometimes misses the point**
  - Prediction Diversity
  - Prediction Context
  - Order of predictions
- **In practice, we care only to predict high ratings:**
  - RMSE might penalize a method that does well for high ratings and badly for others

# References

---

- <https://heartbeat.fritz.ai/recommender-systems-with-python-part-ii-collaborative-filtering-k-nearest-neighbors-algorithm-c8dcd5fd89b2>
  - <https://www.mygreatlearning.com/blog/masterclass-on-movie-recommendation-system/>
  - <https://www.coursera.org/lecture/machine-learning-applications-big-data/recommender-systems-introduction-part-i-9qabY>
  - [https://www.youtube.com/watch?v=1JRrCEgiyHM&list=PLLssT5z\\_DsK9JDLcT8T62VtzwW9LNepV&index=41](https://www.youtube.com/watch?v=1JRrCEgiyHM&list=PLLssT5z_DsK9JDLcT8T62VtzwW9LNepV&index=41)  
<https://web.stanford.edu/class/cs246/>
  - : <http://www.mmds.org>
  - [Getting Started with a Movie Recommendation System | Kaggle](#)
-



---

# Thank you