

Topic Modelling

COMP90042

Natural Language Processing

Lecture 20

Semester 1 2021 Week 10

Jey Han Lau



THE UNIVERSITY OF
MELBOURNE

Making Sense of Text

- English Wikipedia: 6M articles
- Twitter: 500M tweets per day
- New York Times: 15M articles
- arXiv: 1M articles
- What can we do if we want to learn something about these document collections?

Questions

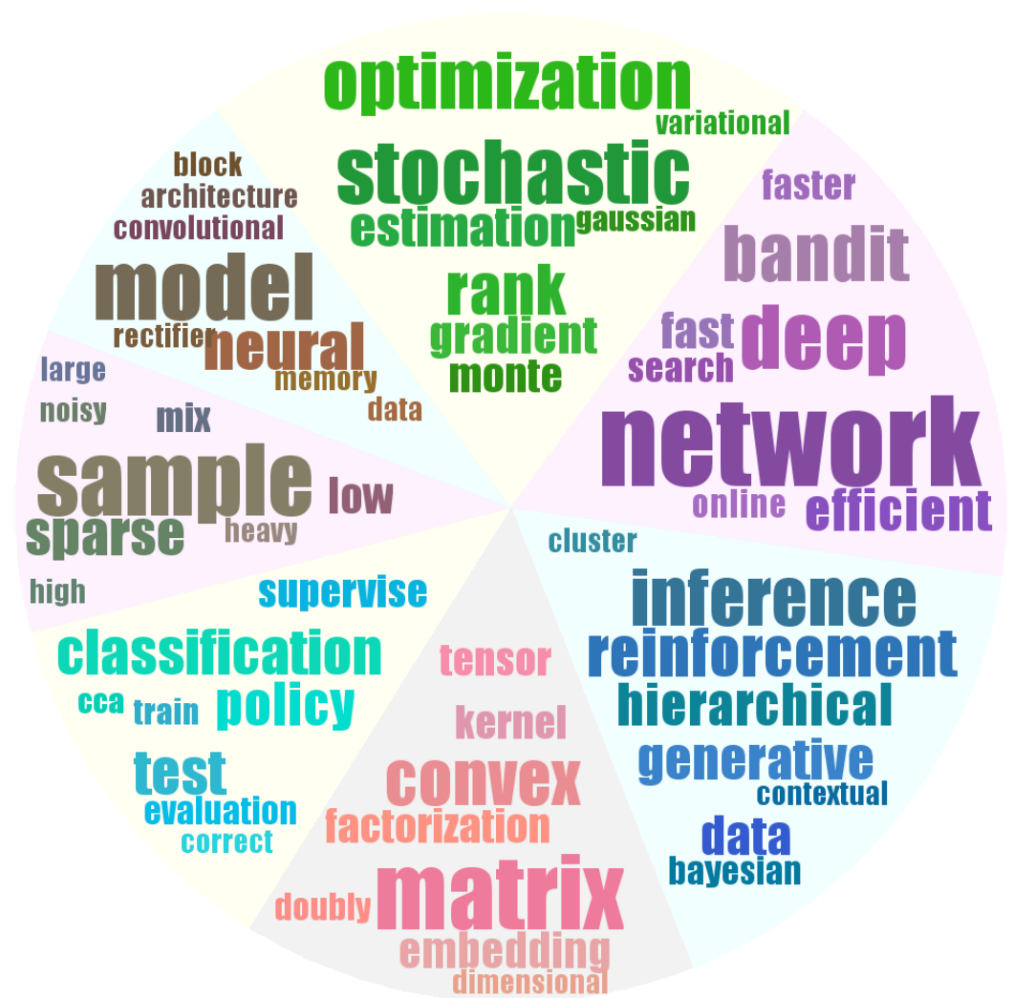
- What are the less popular topics on Wikipedia?
- What are the big trends on Twitter in the past month?
- How do the themes/topics evolve over time in New York Times from 1900s to 2000s?
- What are some influential research areas?

Topic Models To The Rescue

- Topic models learn common, overlapping themes in a document collection
- Unsupervised model
 - ▶ No labels; input is just the documents!
- What's the output of a topic model?
 - ▶ Topics: each topic associated with a list of words
 - ▶ Topic assignments: each document associated with a list of topics

What Do Topics Look Like?

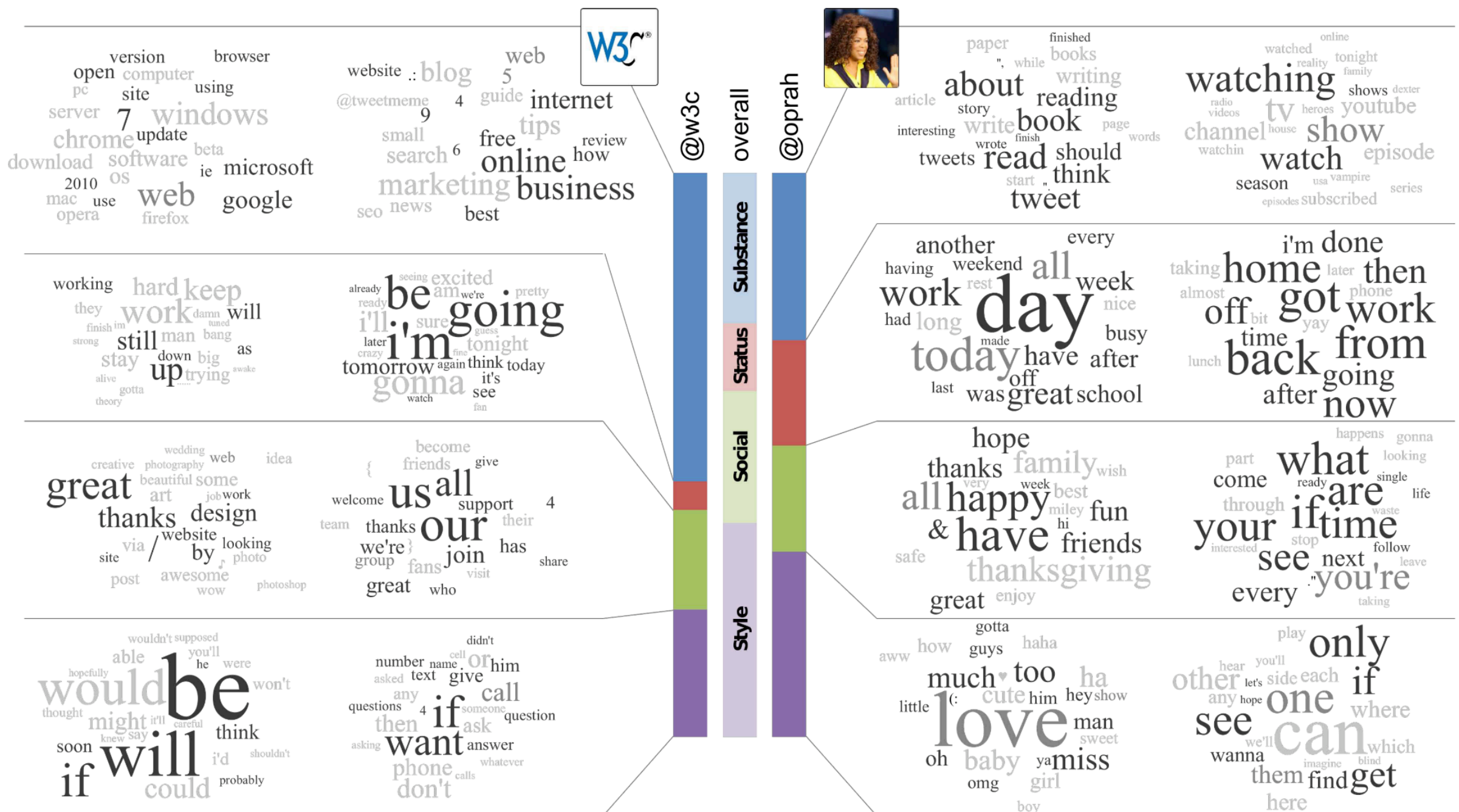
- A list of words
- Collectively describes a concept or subject
- Words of a topic typically appear in the same set of documents in the corpus



Wikipedia Topics



Twitter Topics



New York Times Topics

music
band
songs
rock
album
jazz
pop
song
singer
night

book
life
novel
story
books
man
stories
love
children
family

art
museum
show
exhibition
artist
artists
paintings
painting
century
works

game
knicks
nets
points
team
season
play
games
night
coach

show
film
television
movie
series
says
life
man
character
know

theater
play
production
show
stage
street
broadway
director
musical
directed

clinton
bush
campaign
gore
political
republican
dole
presidential
senator
house

stock
market
percent
fund
investors
funds
companies
stocks
investment
trading

restaurant
sauce
menu
food
dishes
street
dining
dinner
chicken
served

budget
tax
governor
county
mayor
billion
taxes
plan
legislature
fiscal

Applications of topic models?

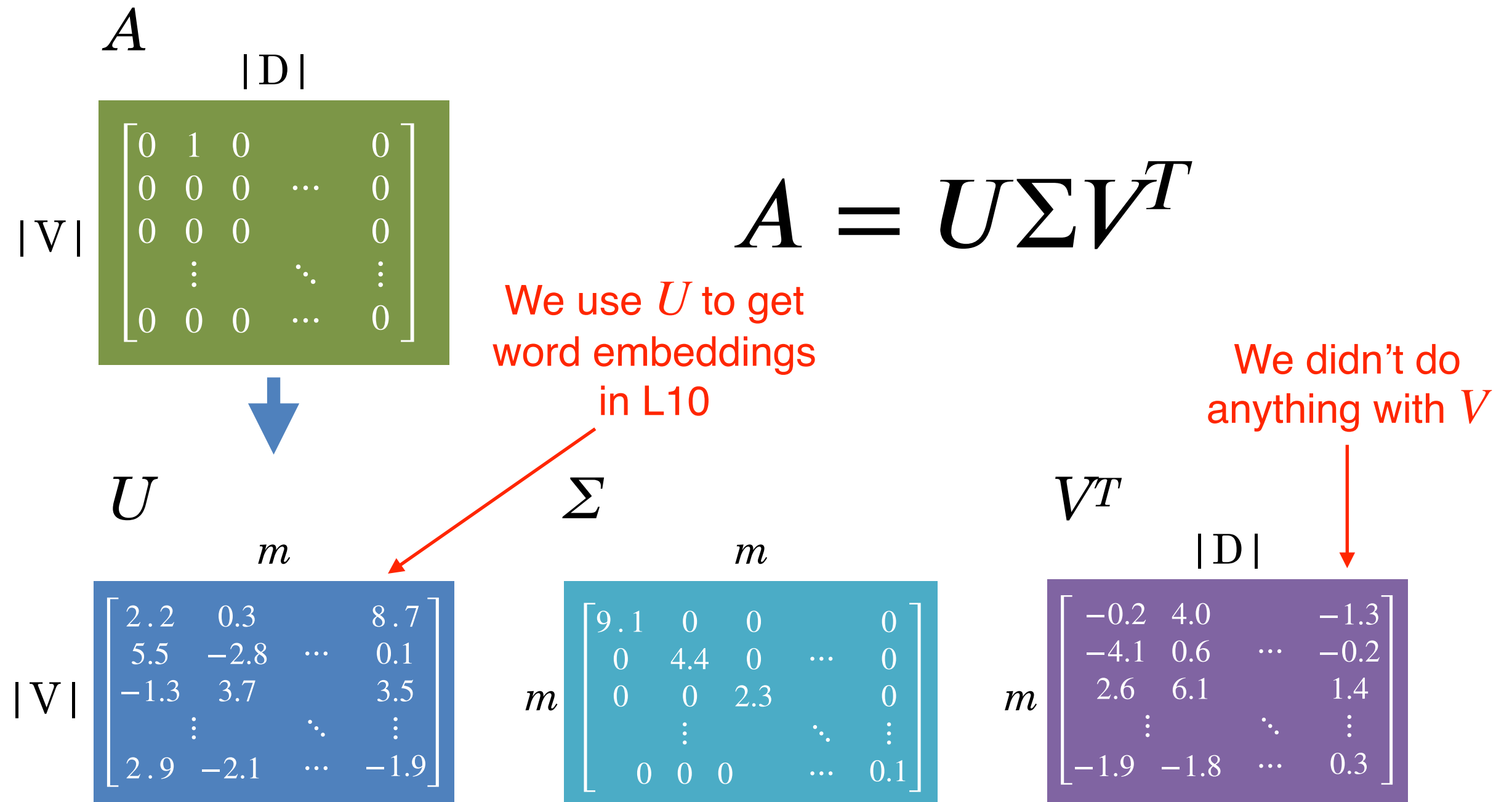
- Personalised advertising
- Search engine
- Discover senses of polysemous words
- Part-of-speech tagging

Outline

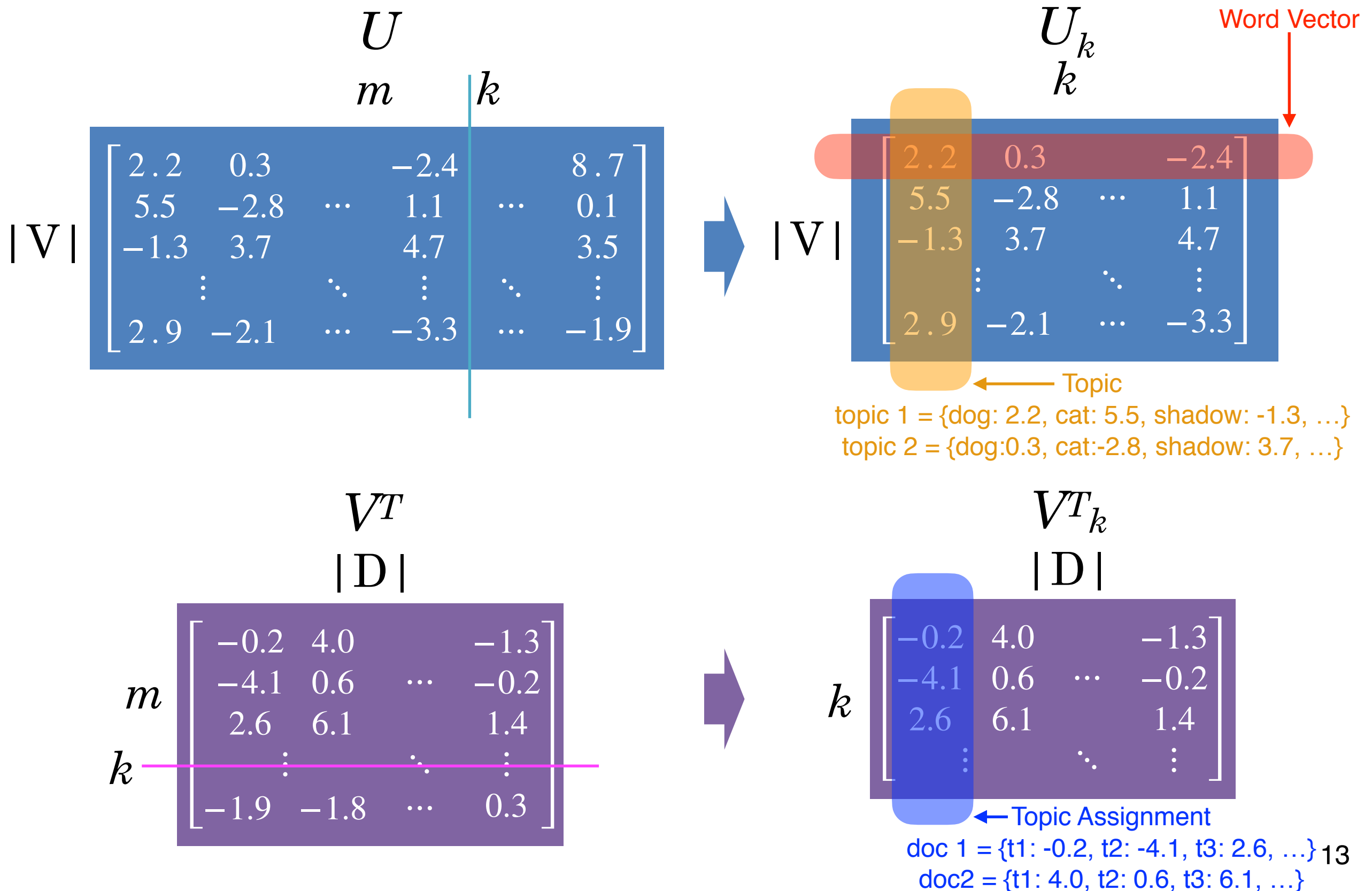
- A Brief History of Topic Models
- Latent Dirichlet Allocation
- Evaluation

A Brief History of Topic Models

Latent Semantic Analysis (L10)

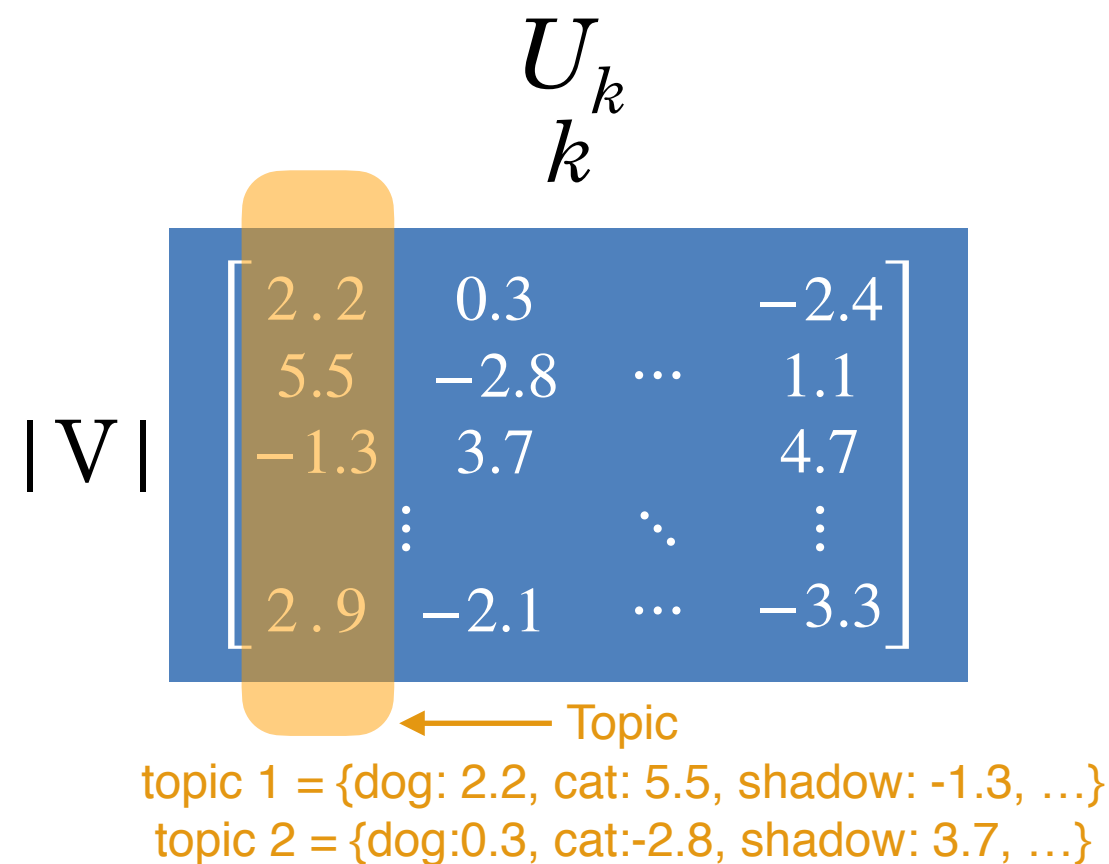


$$A = U\Sigma V^T \quad \text{LSA: Truncate}$$



Issues

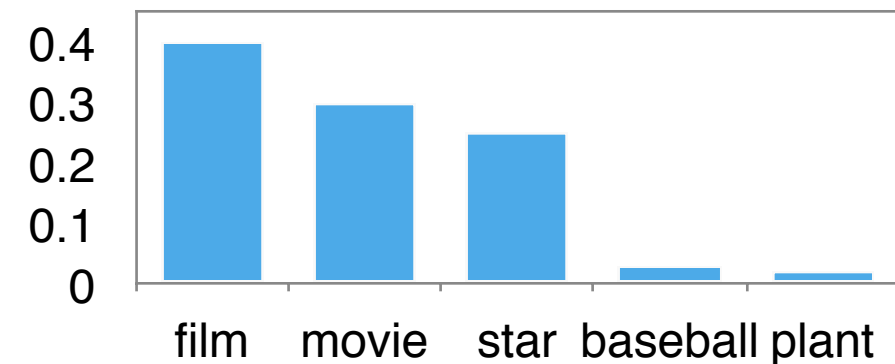
- Positive and negative values in the U and V^T
- Difficult to interpret



Probabilistic LSA

- Based on a probabilistic model

Topic 1

Word distribution
for a topic

$$P(w, d) = P(w | d)P(d)$$

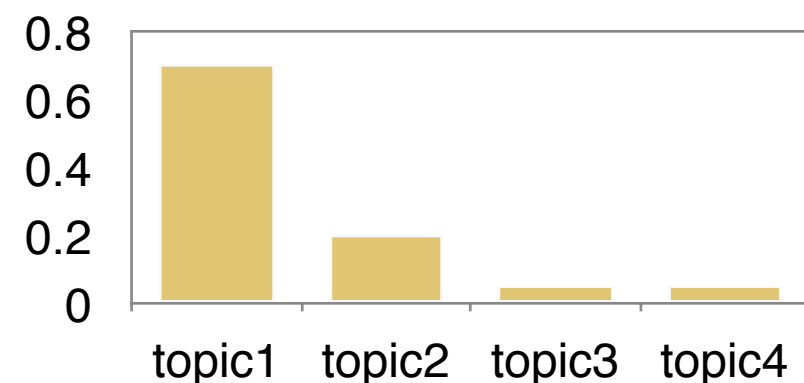
$$= P(d) \sum_T P(w | t)P(t | d)$$

Joint probability of a
word and a document

Number of topics

Topic distribution
for a document

Document 1



Issues

- No more negative values!
- PLSA can learn topics and topic assignment for documents in the train corpus
- But it is unable to infer topic distribution on **new documents**
- PLSA needs to be re-trained for new documents

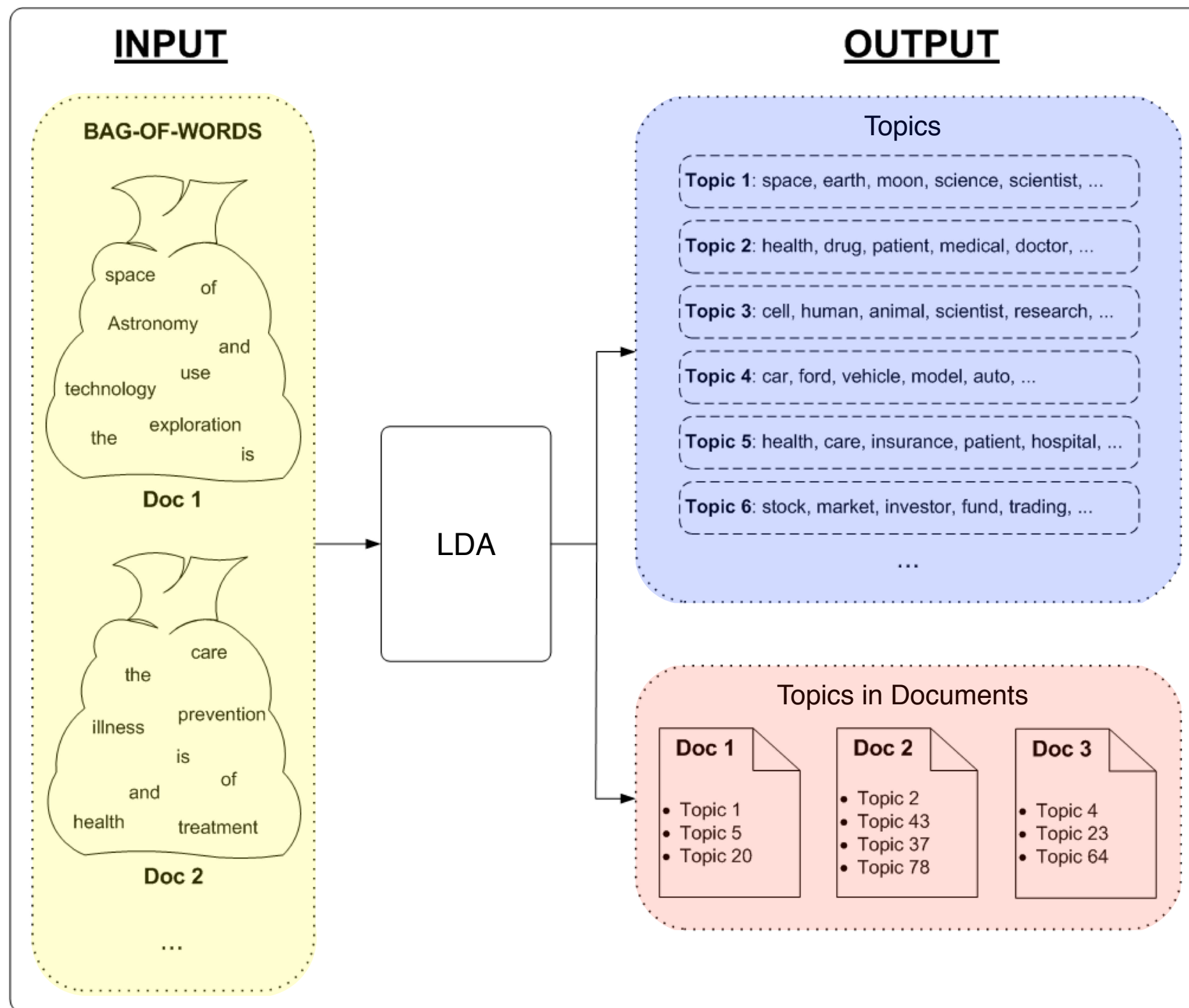
Latent Dirichlet Allocation

- Introduces a prior to the document-topic and topic-word distribution
- Fully generative: trained LDA model can infer topics on unseen documents!
- LDA is a Bayesian version of PLSA

Latent Dirichlet Allocation

LDA

- Core idea: assume each document contains **a mix of topics**
- But the topic structure is **hidden (latent)**
- LDA infers the topic structure given the observed words and documents
- LDA produces soft clusters of documents (based on topic overlap), rather than hard clusters
- Given a trained LDA model, it can infer topics on new documents (not part of train data)

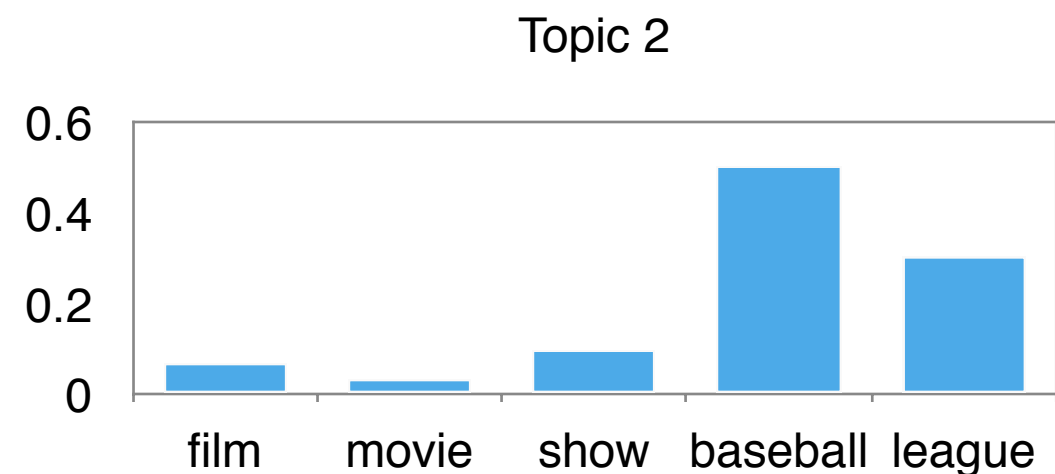
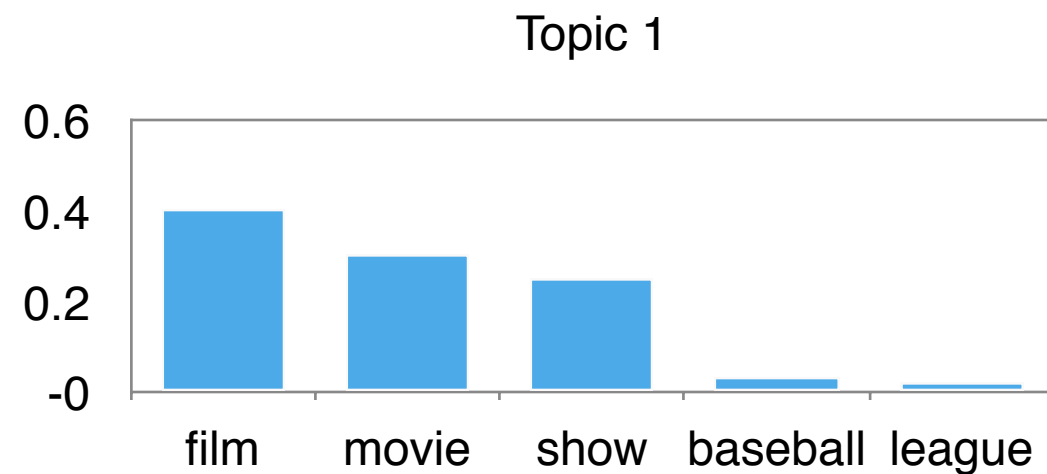


Input

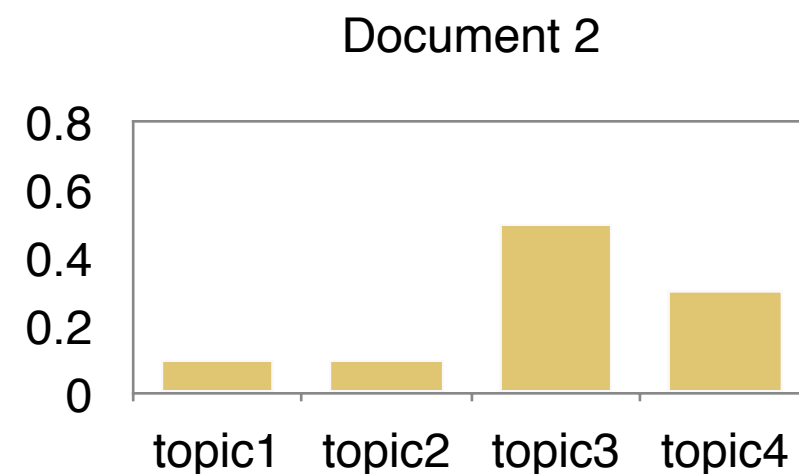
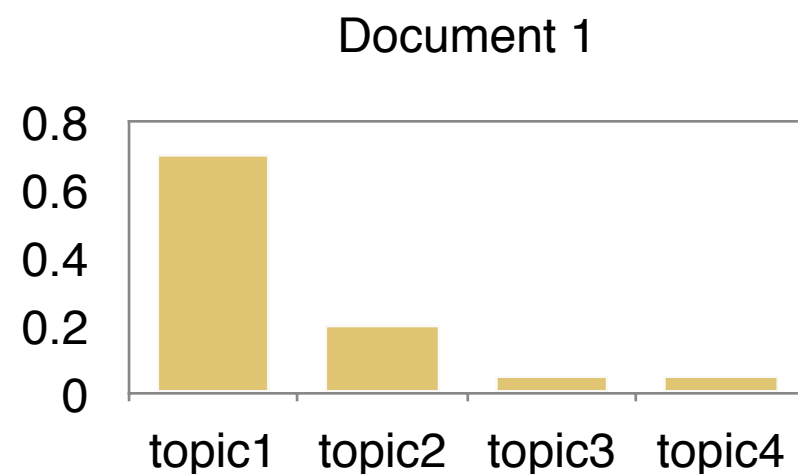
- A collection of documents
- Bag-of-words
- Good preprocessing practice:
 - ▶ Remove stopwords
 - ▶ Remove low and high frequency word types
 - ▶ Lemmatisation

Output

- **Topics:** distribution over words in each topic



- **Topic assignment:** distribution over topics in each document



Learning

- How do we learn the latent topics?
- Two main family of algorithms:
 - ▶ Variational methods
 - ▶ Sampling-based methods

Sampling Method (Gibbs)

1. Randomly assign topics to all tokens in documents

doc₁	mouse: t₁	cat: t₃	rat: t₂	chase: t₁	mouse: t₃
doc₂	scroll: t₁	mouse: t₃	scroll: t₃	scroll: t₂	click: t₂
doc₃	tonight: t₂	baseball: t₁	tv: t₂	exciting: t₁	

2. Collect **topic-word** and **document-topic co-occurrence statistics** based on the assignments

	mouse	cat	scroll	tv	...
t₁	0.01	0.01	0.01	0.01	
t₂	0.01	0.01	0.01	0.01	
t₃	0.01	0.01	0.01	0.01	

	t₁	t₂	t₃
d₁	0.1	0.1	0.1
d₂	0.1	0.1	0.1
...			

Initialise co-occurrence matrix with β (=0.01) and α (=0.1) priors

Sampling Method (Gibbs)

1. Randomly assign topics to all tokens in documents

doc ₁	mouse: ?	cat: t ₃	rat: t ₂	chase: t ₁	mouse: t ₃
doc ₂	scroll: t ₁	mouse: t ₃	scroll: t ₃	scroll: t ₂	click: t ₂
doc ₃	tonight: t ₂	baseball: t ₁	tv: t ₂	exciting: t ₁	

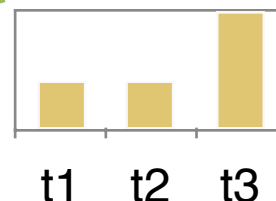
2. Collect **topic-word** and **document-topic co-occurrence statistics** based on the assignments

	mouse	cat	scroll	tv	...
t ₁	1.01 -1	0.01	1.01	0.01	
t ₂	0.01	0.01	1.01	1.01	
t ₃	2.01	1.01	1.01	0.01	

	t ₁	t ₂	t ₃
d ₁	2.1 -1	1.1	2.1
d ₂	1.1	2.1	2.1
...			

3. Go through every word token in corpus and sample a new topic:

$$\triangleright P(t_i | w, d) \propto P(t_i | w)P(t_i | d)$$



Need to de-allocate the current topic assignment and update the co-occurrence matrices before sampling

4. Go to step 2 and repeat until convergence

$$P(t_1 | w, d) = P(t_1 | \text{mouse}) \times P(t_1 | d_1)$$

$$\frac{0.01}{0.01 + 0.01 + 2.01} \times \frac{1.1}{1.1 + 1.1 + 2.1} \quad 25$$

Sampling Method (Gibbs)

1. Randomly assign topics to all tokens in documents

doc ₁	mouse: ?	cat: t ₃	rat: t ₂	chase: t ₁	mouse: t ₃
doc ₂	scroll: t ₁	mouse: t ₃	scroll: t ₃	scroll: t ₂	click: t ₂
doc ₃	tonight: t ₂	baseball: t ₁	tv: t ₂	exciting: t ₁	

2. Collect **topic-word** and **document-topic co-occurrence** statistics based on the assignments

	mouse	cat	scroll	tv	...
t ₁	1.01 -1	0.01	1.01	0.01	
t ₂	0.01	0.01	1.01	1.01	
t ₃	2.01	1.01	1.01	0.01	

	t ₁	t ₂	t ₃
d ₁	2.1 -1	1.1	2.1
d ₂	1.1	2.1	2.1
...			

3. Go through every word token in corpus and sample a new topic:

$$\triangleright P(t_i | w, d) \propto P(t_i | w)P(t_i | d)$$

$$P(t_3 | w, d) = ?$$

4. Go to step 2 and repeat until convergence

$$\frac{2.01}{0.01 + 0.01 + 2.01} \times \frac{2.1}{1.1 + 1.1 + 2.1}$$

When Do We Stop?

- Train until convergence
- Convergence = model probability of training set becomes stable
- How to compute model probability?

$$\log P(w_1, w_2, \dots, w_m) = \log \sum_{j=0}^T P(w_1 | t_j) P(t_j | d_{w_1}) + \dots + \log \sum_{j=0}^T P(w_m | t_j) P(t_j | d_{w_m})$$

► $m = \# \text{word tokens}$

Based on the topic-word
co-occurrence matrix

Based on the document-topic
co-occurrence matrix

Infer Topics For New Documents

1. Randomly assign topics to all tokens in new/test documents

testdoc ₁	tiger: t_2	cow: t_1	cat: t_3	tiger: t_3	
testdoc ₂	football: t_2	live: t_2	men: t_2	fun: t_3	soccer: t_1
testdoc ₃	news: t_1	cnn: t_3	tonight: t_1		

2. Update document-topic matrix based on the assignments; but use the trained topic-word matrix (kept fixed)

from trained topic model

→

	mouse	cat	scroll	tv	...
t_1	5.01	4.01	1.01	0.01	
t_2	0.01	2.01	1.01	8.01	
t_3	0.01	0.01	4.01	0.01	

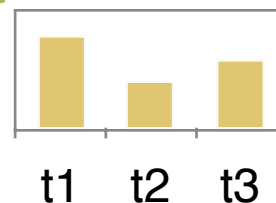
new matrix

←

	t_1	t_2	t_3
td ₁	1.1	1.1	2.1
td ₂	1.1	3.1	1.1
...			

3. Go through every word in the test documents and sample topics:

$$\triangleright P(t_i | w, d) \propto P(t_i | w)P(t_i | d)$$



4. Go to step 2 and repeat until convergence

Hyper-Parameters

- T : number of topic

attorney
death
charges
judge
authorities

economic
economy
company
market
sales

Low T (<10): broad topics

prison judge
manning
attorney
classified
bradley

media
oprah money
winfrey
franken
network

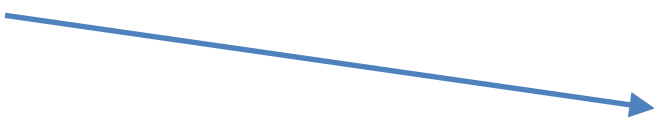
High T (100+): fine-grained, specific topics

Hyper-Parameters

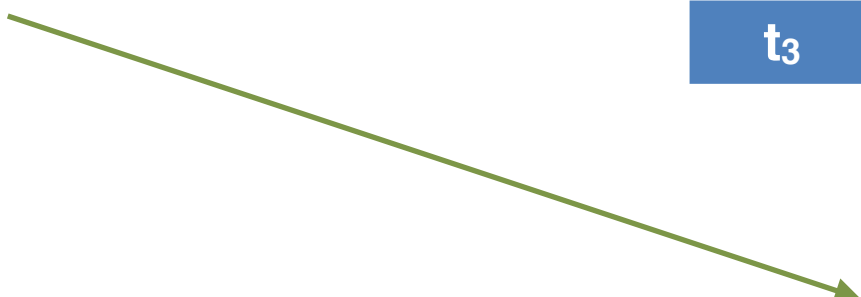
- β : prior on the topic-word distribution
- α : prior on the document-topic distribution
- Analogous to k in add- k smoothing in N -gram LM
- Pseudo counts to initialise co-occurrence matrix:

► $p(w | t): \beta$

► $p(t | d): \alpha$





	mouse	cat	scroll	tv	...
t_1	0.01	0.01	0.01	0.01	
t_2	0.01	0.01	0.01	0.01	
t_3	0.01	0.01	0.01	0.01	



	t_1	t_2	t_3
d_1	0.1	0.1	0.1
d_2	0.1	0.1	0.1
...			

Hyper-Parameters

- High prior values \rightarrow flatter distribution 
- ▶ a very very large value would lead to a uniform distribution
- Low prior values \rightarrow peaky distribution 
- β : generally small (< 0.01)
 - ▶ Large vocabulary, but we want each topic to focus on specific themes
- α : generally larger (> 0.1)
 - ▶ Multiple topics within a document

Evaluation

How To Evaluate Topic Models?

- Unsupervised learning → no labels
- Intrinsic evaluation:
 - ▶ model logprob / perplexity on test documents

$$\log L = \sum_W \sum_T \log P(w | t) P(t | d_w)$$

$$\text{ppl} = \exp^{\frac{-\log L}{W}}$$

Issues with Perplexity

- More topics = better (lower) perplexity
- Smaller vocabulary = better perplexity
 - ▶ Perplexity not comparable for different corpora, or different tokenisation/preprocessing methods
- Does not correlate with human perception of topic quality
- Extrinsic evaluation the way to go:
 - ▶ Evaluate topic models based on downstream task

Topic Coherence

- A better intrinsic evaluation method
- Measure how **coherent** the generated topics

food,
farmers,
rice,
farm,
agriculture

simply
give unionist
choice
count
i.e.

- A good topic model is one that generates more coherent topics

Word Intrusion

- Idea: inject one random word to a topic

{farmers, farm, food, rice, agriculture}



{farmers, farm, food, rice, **cat**, agriculture}

- Ask users to guess which is the **intruder word**
- Correct guess → topic is coherent
- Try guess the intruder word in:
 - ▶ {choice, count, village, i.e., simply, unionist}
- Manual effort; does not scale

PMI \approx Coherence?

- High PMI for a pair of words \rightarrow words are correlated
 - ▶ PMI(farm, rice) \uparrow
 - ▶ PMI(choice, village) \downarrow
- If all word pairs in a topic has high PMI \rightarrow topic is coherent
- If most topics have high PMI \rightarrow good topic model
- Where to get word co-occurrence statistics for PMI?
 - ▶ Can use same corpus for topic model
 - ▶ A better way is to use an external corpus (e.g. Wikipedia)

PMI

- Compute pairwise PMI of top- N words in a topic

$$\text{PMI}(t) = \sum_{j=2}^N \sum_{i=1}^{j-1} \log \frac{P(w_i, w_j)}{P(w_i)P(w_j)}$$

- Given topic: {farmers, farm, food, rice, agriculture}
- **Coherence** = sum PMI for all word pairs:
 - ▶ $\text{PMI}(\text{farmers, farm}) + \text{PMI}(\text{farmers, food})$
 $+ \dots + \text{PMI}(\text{rice, agriculture})$

Variants

- Normalised PMI

$$\text{NPMI}(t) = \sum_{j=2}^N \sum_{i=1}^{j-1} \frac{\log \frac{P(w_i, w_j)}{P(w_i)P(w_j)}}{-\log P(w_i, w_j)}$$

- Conditional probability

$$\text{LCP}(t) = \sum_{j=2}^N \sum_{i=1}^{j-1} \log \frac{P(w_i, w_j)}{P(w_i)}$$

PMI Examples

Topic	PMI	NPMI
cell hormone insulin muscle receptor	0.61	0.59
electron laser magnetic voltage wavelength	0.54	0.52
magnetic neutrino particle quantum universe	0.55	0.55
album band music release song	0.37	0.56
college education school student university	0.38	0.57
city county district population town	0.34	0.52

A Final Word

- Topic model: an unsupervised model for learning latent concepts in a document collection
- LDA: a popular topic model
 - ▶ Learning
 - ▶ Hyper-parameters
- How to evaluate topic models?
 - ▶ Topic coherence