UE 803 - Data Science for NLP

Lecture 12: Topic Modeling

Claire Gardent - CNRS / LORIA

Previous Lecture: Lexical Semantics

- Creating representations to ***capture the meaning of words*** and lexical relations (relatedness, similarity, antonymy) between words
- Word Representation: a vector capturing the common contexts (co-occuring words) in which a word occurs

Today: Topic Modeling

- Creating representations which ***capture what a document is about***
- Alternative to IR representation of documents
    - IR: A document is represented by a vector indicating which ***tokens*** occur in that document
    - Topic Modeling: A document is represented by a vector indicating which ***topics*** are discussed in that document

Topic Modeling

Aims to identify (latent) topics in a text corpus

- What are the topics discussed in the corpus ?
- Which document discusses which topic ?
- ***Dimensionality Reduction*** : "Reduces" a document to a set of topics

Unsupervised

- Learns from raw data. No labels or classes are given

Several algorithms can be used to perform topic modeling.

- Latent Semantic Analysis (LSA)
- Non Negative Matrix Factorization (NMF)
- Latent Dirichlet Allocation (LDA)

Python Libraries

Scikit-Learn

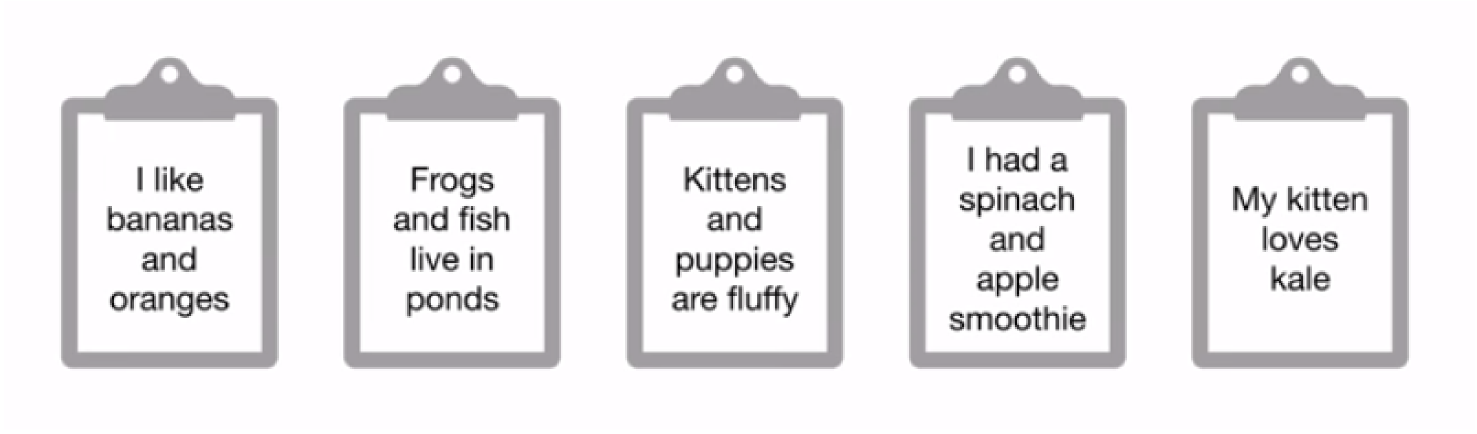
Gensim

- Gensim = “Generate Similar”
- Unsupervised semantic modelling from plain text
- Tutorials

pyLDAvis

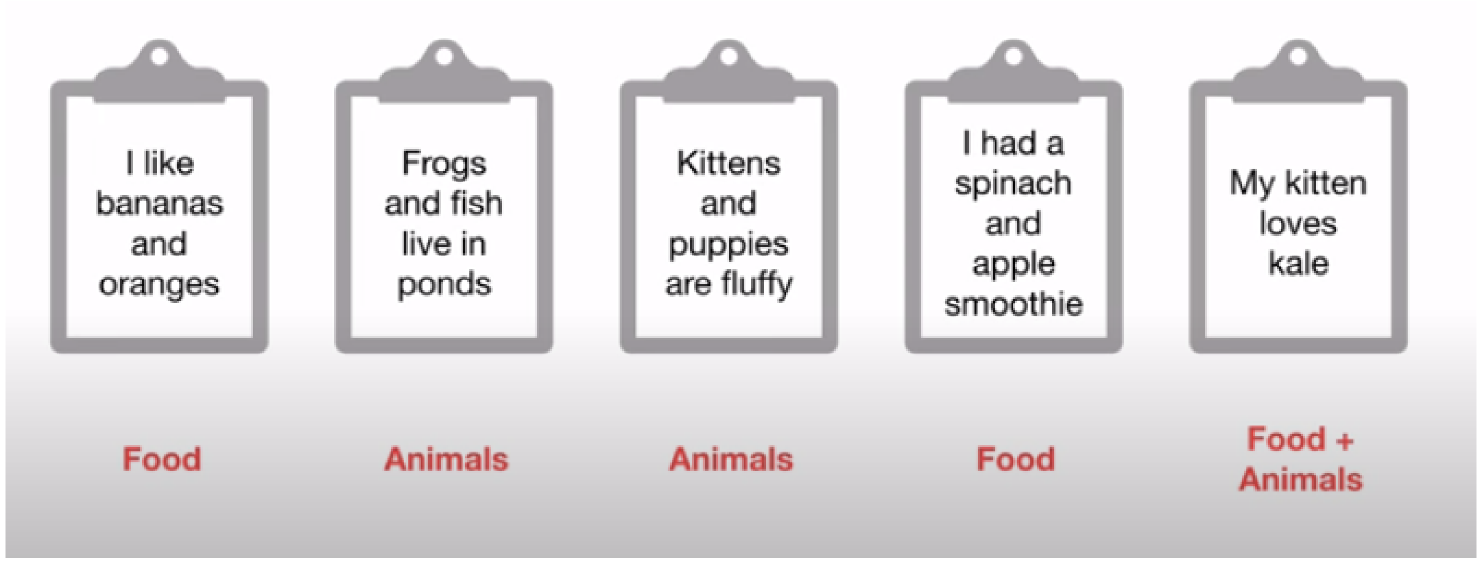
- Python library for interactive topic model visualization.
- pip install pyldavis

A Simple Example (LDA View)


I like bananas and oranges
Frogs and fish live in ponds
Kittens and puppies are fluffy
I had a spinach and apple smoothie
My kitten loves kale

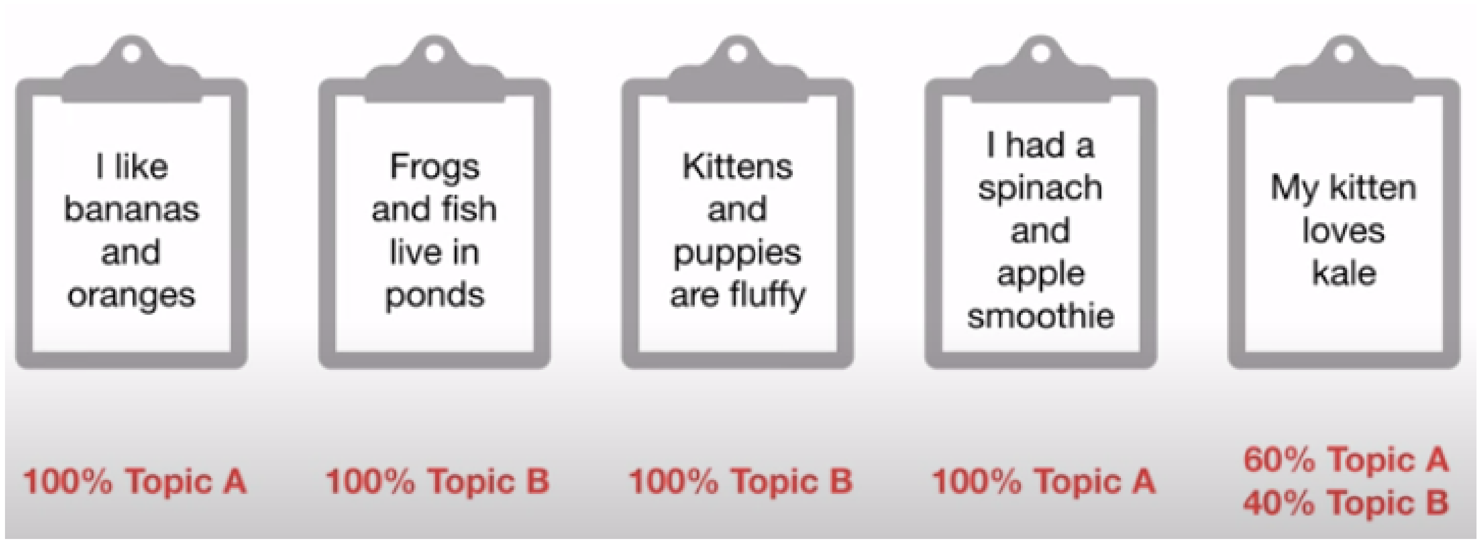
- Can we group these documents according to the topics they address ?
- What are those topics ?
- Which words are associated with each topic ?
- N.B. we have no labels, just raw text. Hence the task is ***unsupervised***$_{\text{src}}$

A Simple Example (LDA view)


I like bananas and oranges
Frogs and fish live in ponds
Kittens and puppies are fluffy
I had a spinach and apple smoothie
My kitten loves kale
Food
Animals
Animals
Food
Food + Animals

- Intuitively, the documents are about two topics: food vs animals
- Some documents are about one topic only
- Other documents are about both topics

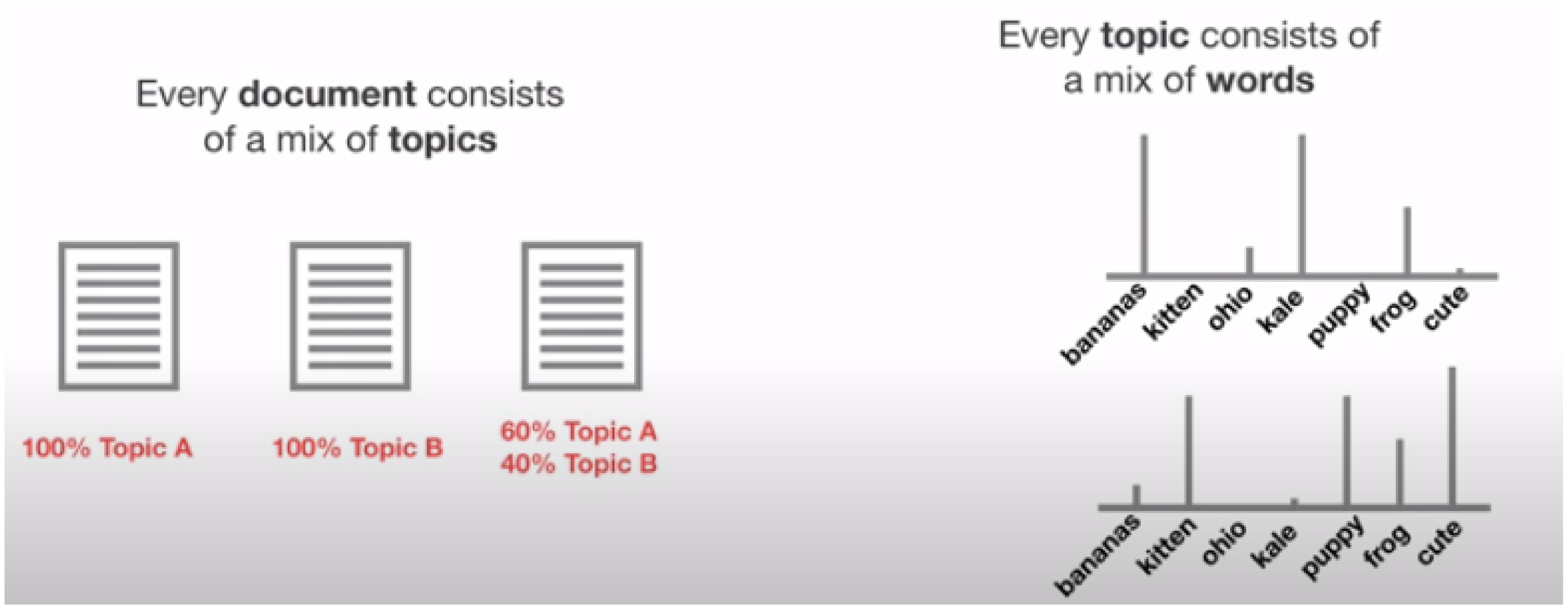
A Simple Example (LDA view)


I like bananas and oranges
100% Topic A
Frogs and fish live in ponds
100% Topic B
Kittens and puppies are fluffy
100% Topic B
I had a spinach and apple smoothie
100% Topic A
My kitten loves kale
60% Topic A
40% Topic B

A document is about a mix of topics

A topic is a mix of words

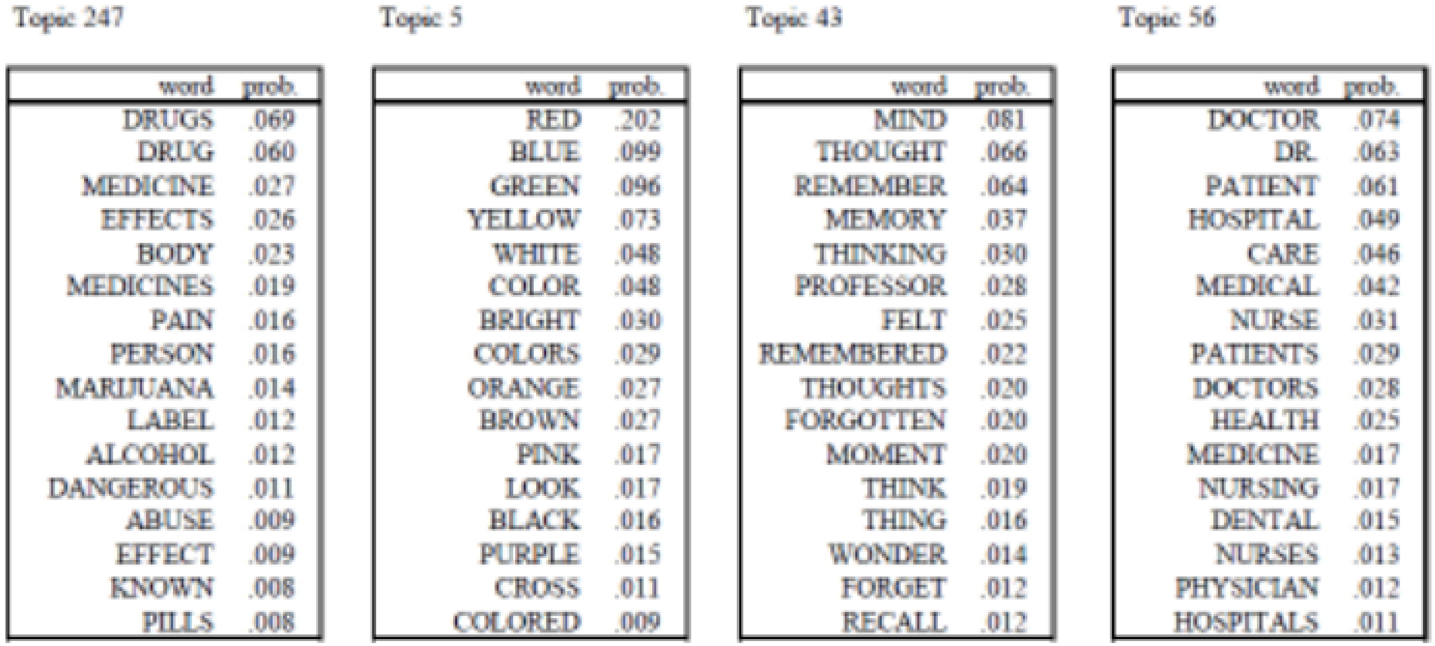
A Simple Example (LDA view)



Topic modelling can associate a document with ***multiple topics*** (different from classification where each document is mapped to a single class)

A Real Life Example

- Touchstone Applied Science Associates Corpus
- 60,527 documents from 6,333 textbooks, works of literature, and popular works of fiction and nonfiction.



Topic 247

word	prob.
DRUGS	.069
DRUG	.060
MEDICINE	.027
EFFECTS	.026
BODY	.023
MEDICINES	.019
PAIN	.016
PERSON	.016
MARIJUANA	.014
LABEL	.012
ALCOHOL	.012
DANGEROUS	.011
ABUSE	.009
EFFECT	.009
KNOWN	.008
PILLS	.008

Topic 5

word	prob.
RED	.202
BLUE	.099
GREEN	.096
YELLOW	.073
WHITE	.048
COLOR	.048
BRIGHT	.030
COLORS	.029
ORANGE	.027
BROWN	.027
PINK	.017
LOOK	.017
BLACK	.016
PURPLE	.015
CROSS	.011
COLORED	.009

Topic 43

word	prob.
MIND	.081
THOUGHT	.066
REMEMBER	.064
MEMORY	.037
THINKING	.030
PROFESSOR	.028
FELT	.025
REMEMBERED	.022
THOUGHTS	.020
FORGOTTEN	.020
MOMENT	.020
THINK	.019
THING	.016
WONDER	.014
FORGET	.012
RECALL	.012

Topic 56

word	prob.
DOCTOR	.074
DR.	.063
PATIENT	.061
HOSPITAL	.049
CARE	.046
MEDICAL	.042
NURSE	.031
PATIENTS	.029
DOCTORS	.028
HEALTH	.025
MEDICINE	.017
NURSING	.017
DENTAL	.015
NURSES	.013
PHYSICIAN	.012
HOSPITALS	.011

src

What is Topic Modeling useful for ?

- To identify the topics discussed in a corpus
  We can create ***a graph visualisation of a corpus*** where topics are nodes, edges indicate similar topics and topics are labelled with the most relevant words for that topic

- To group together similar documents which discuss the same topics

- To compare the topics discussed in different corpora

More generally:

- To ***discover*** hidden topics

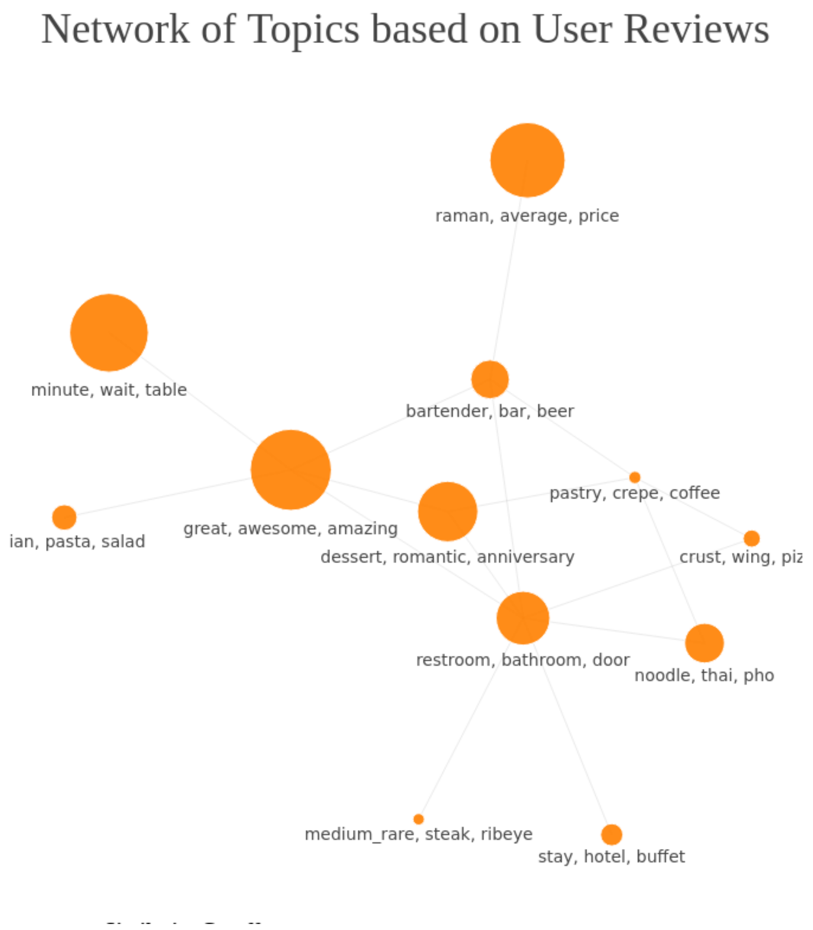
- To ***annotate*** documents with these topics

- To use these topic annotations to ***organize and search texts collections*** .

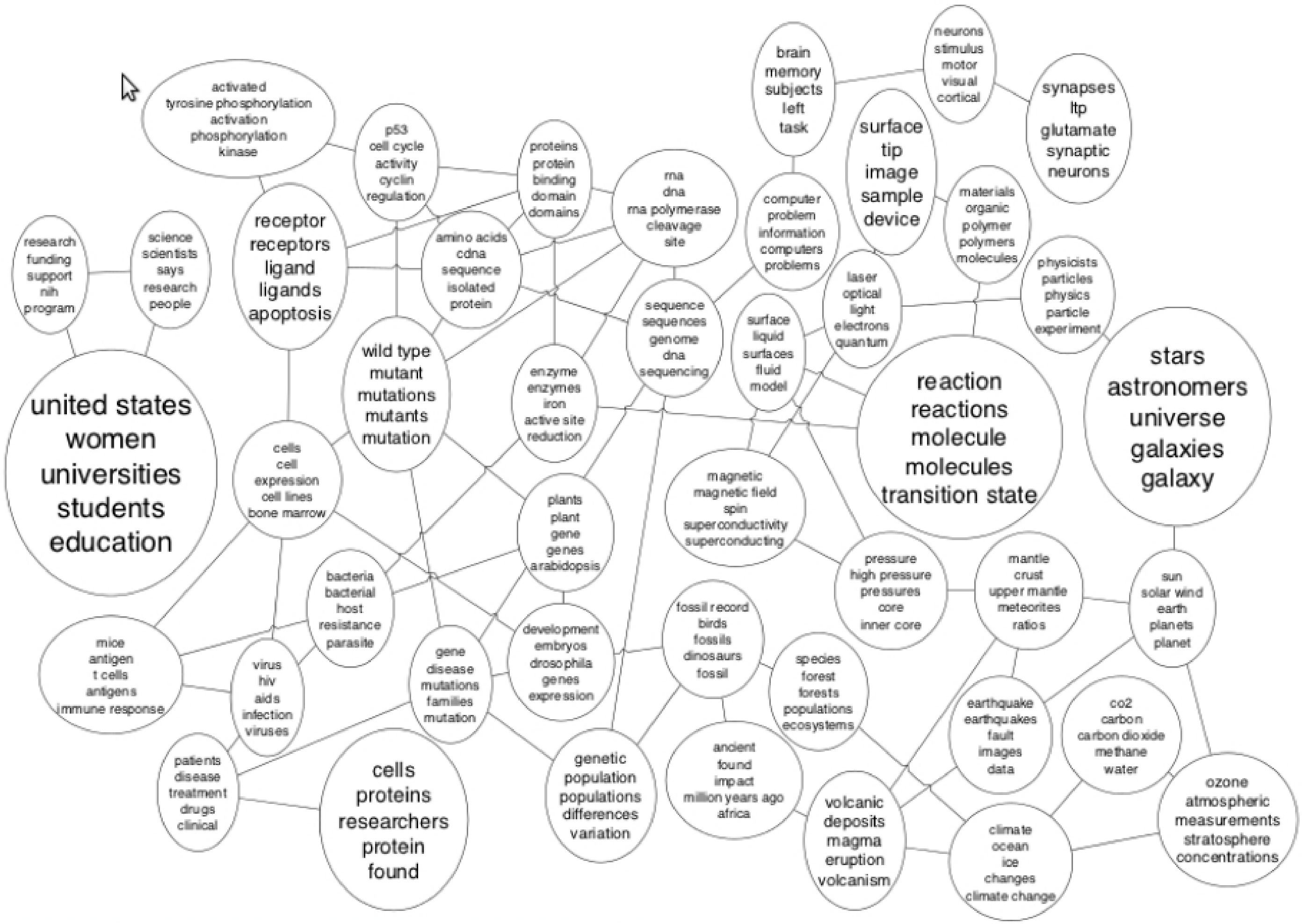
Example Visualisation of the Topics discussed in a Corpus (Using LDA)

23,700 restaurant reviews

- Nodes = topics
- Node size = Nb of tokens present in a topic divided by the total Nb of tokens in the entire text corpus.
- Node labels = top words for each topic (most relevant words, words with highest probability)
- Edges = 2 nodes are linked if their similarity is over the given threshold



Another Topic Graph Example


activated tyrosine phosphorylation activation phosphorylation kinase
p53 cell cycle activity cyclin regulation
proteins protein binding domain domains
rna dna rna polymerase cleavage site
brain memory subjects left task
neurons stimulus motor visual cortical
synapses ltp glutamate synaptic neurons
surface tip image sample device
computer problem information computers problems
materials organic polymer polymers molecules
research funding support nih program
science scientists says research people
receptor receptors ligand ligands apoptosis
amino acids cdna sequence isolated protein
sequence sequences genome dna sequencing
surface liquid surfaces fluid model
laser optical light electrons quantum
physicists particles physics particle experiment
stars astronomers universe galaxies galaxy
united states women universities students education
wild type mutant mutations mutants mutation
enzyme enzymes iron active site reduction
reaction reactions molecule molecules transition state
cells cell expression cell lines bone marrow
magnetic magnetic field spin superconductivity superconducting
plants plant gene genes arabidopsis
bacteria bacterial host resistance parasite
pressure high pressure pressures core inner core
mantle crust upper mantle meteorites ratios
sun solar wind earth planets planet
fossil record birds fossils dinosaurs fossil
development embryos drosophila genes expression
mice antigen t cells antigens immune response
virus hiv aids infection viruses
gene disease mutations families mutation
species forest forests populations ecosystems
earthquake earthquakes fault images data
co2 carbon carbon dioxide methane water
patients disease treatment drugs clinical
cells proteins researchers protein found
genetic population populations differences variation
ancient found impact million years ago africa
volcanic deposits magma eruption volcanism
climate ocean ice changes climate change
ozone atmospheric measurements stratosphere concentrations

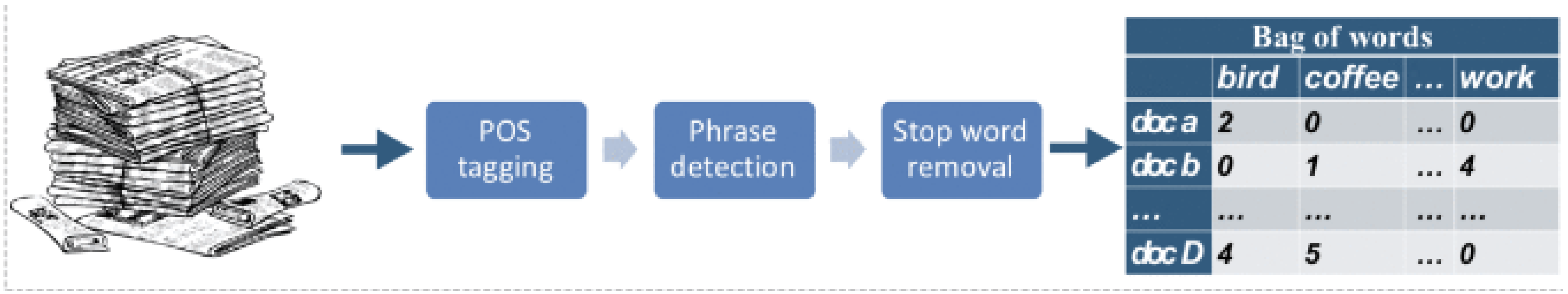
Topic Models

Inferring a Topic Model from a Corpus

1. Gather a corpus (a collection of texts)
2. Specify the basic units (tweet, paragraph, article, document etc.)
3. Text processing: Remove noise (Garbage-in, garbage-out)
4. Encode the corpus into a document/tokens matrix
5. Choose a method for topic/modeling: LSA, LDA, SVD ...
6. Train topic model
7. Interpret the results
    - label the topics
    - quantitative evaluation
    - Visualise e.g., Wordcloud

Representing (encoding) Documents

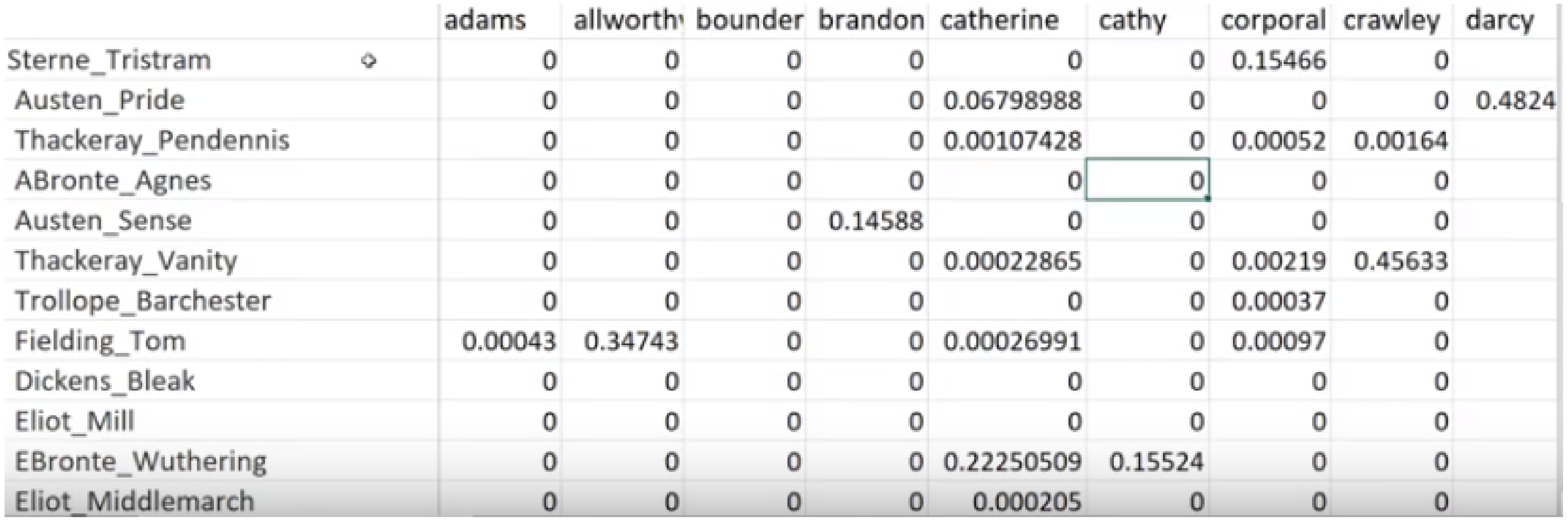
- Documents are represented by a ***bag of words***, the words (tokens) they contain
- Order, syntax is ***not*** taken into account
- BOWs are encoded as vectors
    - whose dimensions are the tokens occurring in the corpus
    - whose size is the size of the corpus vocabulary
    - whose components can be binary (presence/absence of the token in the document), frequencies, tf-idf scores ...



Bag of words				
	bird	*coffee*	...	*work*
doc a	*2*	*0*	...	*0*
doc b	*0*	*1*	...	*4*
...	...	...	...	...
doc D	*4*	*5*	...	*0*

BOW Corpus Representation

- A corpus is represented by a ***Document/token*** matrix
- Each row represents a document
- The columns indicate which words occur in which document



	adams	allworthy	bounder	brandon	catherine	cathy	corporal	crawley	darcy
Sterne_Tristram	0	0	0	0	0	0	0.15466	0	
Austen_Pride	0	0	0	0	0.06798988	0	0	0	0.4824
Thackeray_Pendennis	0	0	0	0	0.00107428	0	0.00052	0.00164	
ABronte_Agnes	0	0	0	0	0	0	0	0	
Austen_Sense	0	0	0	0.14588	0	0	0	0	
Thackeray_Vanity	0	0	0	0	0.00022865	0	0.00219	0.45633	
Trollope_Barchester	0	0	0	0	0	0	0.00037	0	
Fielding_Tom	0.00043	0.34743	0	0	0.00026991	0	0.00097	0	
Dickens_Bleak	0	0	0	0	0	0	0	0	
Eliot_Mill	0	0	0	0	0	0	0	0	
EBronte_Wuthering	0	0	0	0	0.22250509	0.15524	0	0	
Eliot_Middlemarch	0	0	0	0	0.000205	0	0	0	

From Tokens to Topics

- A Document/Token vector has the size of the corpus vocabulary
- This is usually ***very large***
- ***Topic modeling reduces this vocabulary to a set of topics***
- A document is then represented by ***a vector/ a distribution/mix of topics***
- This permits reducing the size of the document vectors
  A document vector now has the size of the set of topics
- It also permits a more abstract view of the content of a document
  Typically, a topic groups together related words

Latent Semantic Analysis (LSA)

and

Singular Value Decomposition (SVD)

Latent Semantic Analysis

- Starts from a Documents/Tokens Matrix A
- Uses Singular Value Decomposition (SVD) to decompose A into a product of three matrices

$$A = U \times \Sigma \times V^\top$$

- This permits reducing the initial set of tokens to a set of topics

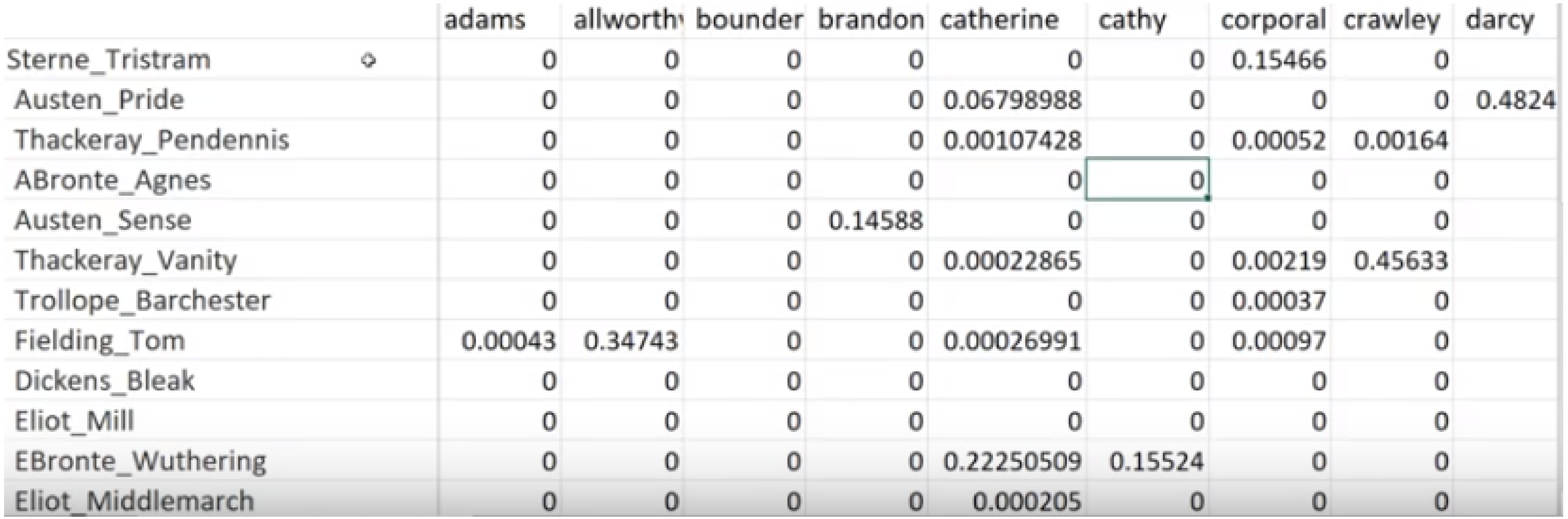
Singular Value Decomposition (SVD)

$$A_{(m,n)} = U_{(m,t)} \times \Sigma_{(t,t)} \times V^\top_{(t,n)}$$

- $A_{(m,n)}$: Documents / Tokens Matrix
  m: nb of documents, n: nb of tokens
- $U_{(m,t)}$: Documents / Topics Matrix
  Document = distribution over topics (mix of topics)
- $S_{(t,t)}$: Singular Value Matrix. Diagonal matrix
  Relative importance of each factor
- $V^\top_{(t,n)}$: Topics / Tokens Matrix
  Topic = distribution over words (mix of words)

Document/Tokens Matrix

Documents = Books

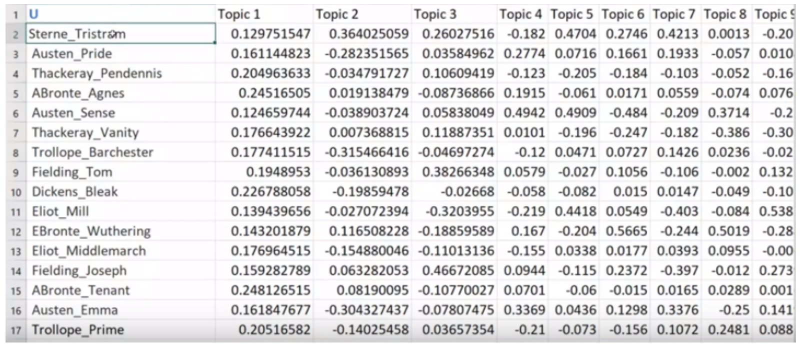


	adams	allworthy	bounder	brandon	catherine	cathy	corporal	crawley	darcy
Sterne_Tristram	0	0	0	0	0	0	0.15466	0	
Austen_Pride	0	0	0	0	0.06798988	0	0	0	0.4824
Thackeray_Pendennis	0	0	0	0	0.00107428	0	0.00052	0.00164	
ABronte_Agnes	0	0	0	0	0	0	0	0	
Austen_Sense	0	0	0	0.14588	0	0	0	0	
Thackeray_Vanity	0	0	0	0	0.00022865	0	0.00219	0.45633	
Trollope_Barchester	0	0	0	0	0	0	0.00037	0	
Fielding_Tom	0.00043	0.34743	0	0	0.00026991	0	0.00097	0	
Dickens_Bleak	0	0	0	0	0	0	0	0	
Eliot_Mill	0	0	0	0	0	0	0	0	
EBronte_Wuthering	0	0	0	0	0.22250509	0.15524	0	0	
Eliot_Middlemarch	0	0	0	0	0.000205	0	0	0	

SVD Decomposition

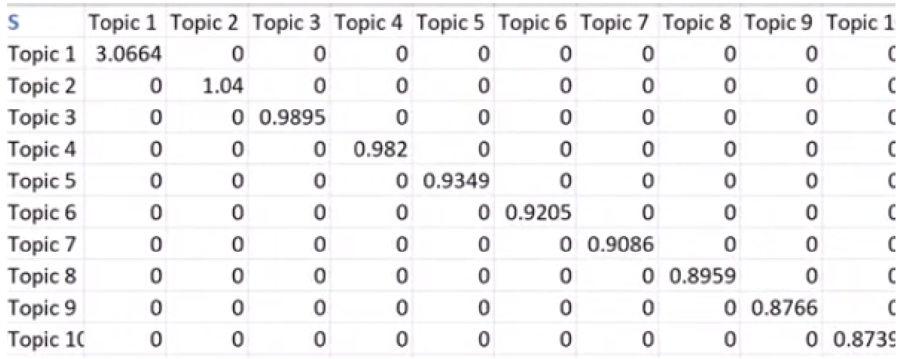
$$A = U \times \Sigma \times V^{\top}$$

$U_{(m,t)}$**: Documents / Topics Matrix**



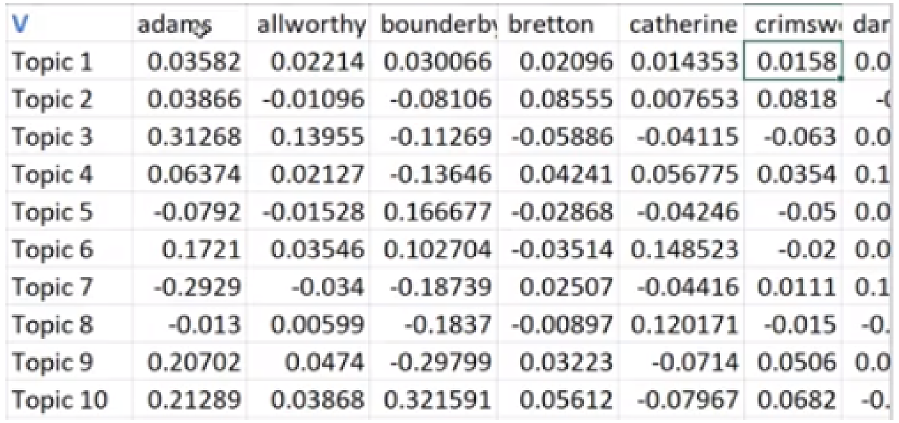
U	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9
Sterne_Tristram	0.129751547	0.364025059	0.26027516	-0.182	0.4704	0.2746	0.4213	0.0013	-0.20
Austen_Pride	0.161144823	-0.282351565	0.03584962	0.2774	0.0716	0.1661	0.1933	-0.057	0.010
Thackeray_Pendennis	0.204963633	-0.034791727	0.10609419	-0.123	-0.205	-0.184	-0.103	-0.052	-0.16
ABronte_Agnes	0.24516505	0.019138479	-0.08736866	0.1915	-0.061	0.0171	0.0559	-0.074	0.076
Austen_Sense	0.124659744	-0.038903724	0.05838049	0.4942	0.4909	-0.484	-0.209	0.3714	-0.2
Thackeray_Vanity	0.176643922	0.007368815	0.11887351	0.0101	-0.196	-0.247	-0.182	-0.386	-0.30
Trollope_Barchester	0.177411515	-0.315466416	-0.04697274	-0.12	0.0471	0.0727	0.1426	0.0236	-0.02
Fielding_Tom	0.1948953	-0.036130893	0.38266348	0.0579	-0.027	0.1056	-0.106	-0.002	0.132
Dickens_Bleak	0.226788058	-0.19859478	-0.02668	-0.058	-0.082	0.015	0.0147	-0.049	-0.10
Eliot_Mill	0.139439656	-0.027072394	-0.3203955	-0.219	0.4418	0.0549	-0.403	-0.084	0.538
EBronte_Wuthering	0.143201879	0.116508228	-0.18859589	0.167	-0.204	0.5665	-0.244	0.5019	-0.28
Eliot_Middlemarch	0.176964515	-0.154880046	-0.11013136	-0.155	0.0338	0.0177	0.0393	0.0955	-0.00
Fielding_Joseph	0.159282789	0.063282053	0.46672085	0.0944	-0.115	0.2372	-0.397	-0.012	0.273
ABronte_Tenant	0.248126515	0.08190095	-0.10770027	0.0701	-0.06	-0.015	0.0165	0.0289	0.001
Austen_Emma	0.161847677	-0.304327437	-0.07807475	0.3369	0.0436	0.1298	0.3376	-0.25	0.141
Trollope_Prime	0.20516582	-0.14025458	0.03657354	-0.21	-0.073	-0.156	0.1072	0.2481	0.088

$\Sigma_{(t,t)}$**: Singular Value Matrix**



S	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	Topic 1
Topic 1	3.0664	0	0	0	0	0	0	0	0	0
Topic 2	0	1.04	0	0	0	0	0	0	0	0
Topic 3	0	0	0.9895	0	0	0	0	0	0	0
Topic 4	0	0	0	0.982	0	0	0	0	0	0
Topic 5	0	0	0	0	0.9349	0	0	0	0	0
Topic 6	0	0	0	0	0	0.9205	0	0	0	0
Topic 7	0	0	0	0	0	0	0.9086	0	0	0
Topic 8	0	0	0	0	0	0	0	0.8959	0	0
Topic 9	0	0	0	0	0	0	0	0	0.8766	0
Topic 1	0	0	0	0	0	0	0	0	0	0.873

$V^{\top}_{(t,n)}$**: Topics / Tokens Matrix**



V	adams	allworthy	bounderby	bretton	catherine	crimsw	dar
Topic 1	0.03582	0.02214	0.030066	0.02096	0.014353	0.0158	0.0
Topic 2	0.03866	-0.01096	-0.08106	0.08555	0.007653	0.0818	-
Topic 3	0.31268	0.13955	-0.11269	-0.05886	-0.04115	-0.063	0.0
Topic 4	0.06374	0.02127	-0.13646	0.04241	0.056775	0.0354	0.1
Topic 5	-0.0792	-0.01528	0.166677	-0.02868	-0.04246	-0.05	0.0
Topic 6	0.1721	0.03546	0.102704	-0.03514	0.148523	-0.02	0.0
Topic 7	-0.2929	-0.034	-0.18739	0.02507	-0.04416	0.0111	0.1
Topic 8	-0.013	0.00599	-0.1837	-0.00897	0.120171	-0.015	-0.
Topic 9	0.20702	0.0474	-0.29799	0.03223	-0.0714	0.0506	0.0
Topic 10	0.21289	0.03868	0.321591	0.05612	-0.07967	0.0682	-0.

The diagonal matrix Σ specifies the relative importance of each factor

Dimensionality Reduction

Low-Rank Approximation

- SVD can be used to compute optimal low-rank approximations
- This is done by setting the smallest of the singular values (in Σ) to zero.
- Once we do that, we can also eliminate the corresponding columns of U and rows in V
- Now ***the number of topics (columns in*** U***) is much smaller***

Dimensionality Reduction

Without reduction

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix}}_{M'} =$$

$$\underbrace{\begin{bmatrix} .13 & .02 & -.01 \\ .41 & .07 & -.03 \\ .55 & .09 & -.04 \\ .68 & .11 & -.05 \\ .15 & -.59 & .65 \\ .07 & -.73 & -.67 \\ .07 & -.29 & .32 \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} .56 & .59 & .56 & .09 & .09 \\ .12 & -.02 & .12 & -.69 & -.69 \\ .40 & -.80 & .40 & .09 & .09 \end{bmatrix}}_{V^{\mathrm{T}}}$$

With reduction

$$\begin{bmatrix} .13 & .02 \\ .41 & .07 \\ .55 & .09 \\ .68 & .11 \\ .15 & -.59 \\ .07 & -.73 \\ .07 & -.29 \end{bmatrix} \begin{bmatrix} 12.4 & 0 \\ 0 & 9.5 \end{bmatrix} \begin{bmatrix} .56 & .59 & .56 & .09 & .09 \\ .12 & -.02 & .12 & -.69 & -.69 \end{bmatrix}$$

$$= \begin{bmatrix} 0.93 & 0.95 & 0.93 & .014 & .014 \\ 2.93 & 2.99 & 2.93 & .000 & .000 \\ 3.92 & 4.01 & 3.92 & .026 & .026 \\ 4.84 & 4.96 & 4.84 & .040 & .040 \\ 0.37 & 1.21 & 0.37 & 4.04 & 4.04 \\ 0.35 & 0.65 & 0.35 & 4.87 & 4.87 \\ 0.16 & 0.57 & 0.16 & 1.98 & 1.98 \end{bmatrix}$$

- Fewer topics: $3 \to 2$
- The new matrix is similar to the initial matrix

Truncated SVD

SVD

- Exact decomposition
- The created matrices fully cover the original matrix

Truncated SVD

- Approximate
- Much faster than SVD
- Size of diagonal matrix must be specified

Sklearn truncated SVD

Non Negative Matrix Factorization

]

SVD

- Values can be negative
- Decomposes a matrix into a product of three matrices

$$A = U \times \Sigma \times V^\top$$

  - Exact but computationally intensive (slow)

Non Negative Matrix Factorisation (NMF)

- No negative values
- Decomposes a matrix into a product of two matrices
  - W: Document / Topics
  - H: Topics / Tokens

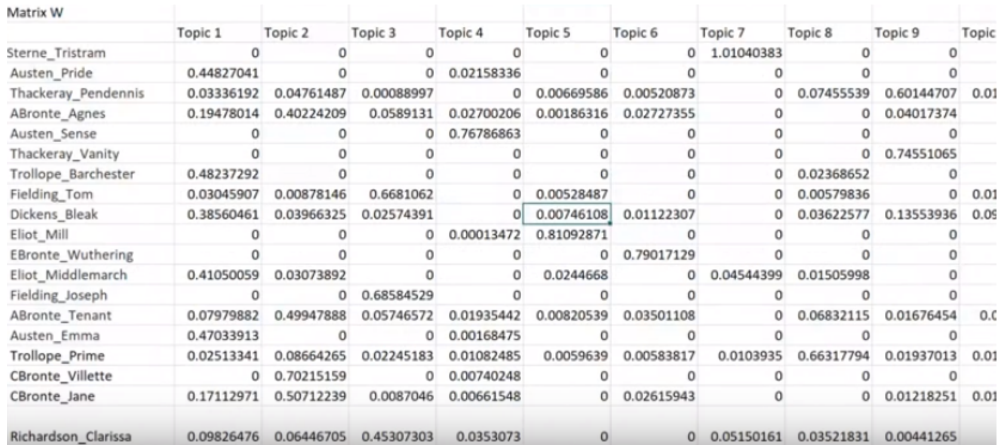
$$A = W \times H$$

- Approximate but fast

sklearn NMF decomposition, NMF Tutorial

NMF Output

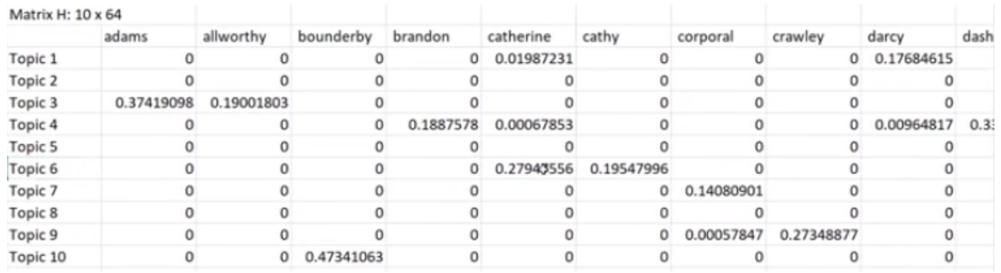
W: Documents / Topics Matrix



Matrix W

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	Topic
Sterne_Tristram	0	0	0	0	0	0	1.01040383	0	0	
Austen_Pride	0.44827041	0	0	0.02158336	0	0	0	0	0	
Thackeray_Pendennis	0.03336192	0.04761487	0.00088997	0	0.00669586	0.00520873	0	0.07455539	0.60144707	0.01
ABronte_Agnes	0.19478014	0.40224209	0.0589131	0.02700206	0.00186316	0.02727355	0	0	0.04017374	
Austen_Sense	0	0	0	0.76786863	0	0	0	0	0	
Thackeray_Vanity	0	0	0	0	0	0	0	0	0.74551065	
Trollope_Barchester	0.48237292	0	0	0	0	0	0	0.02368652	0	
Fielding_Tom	0.03045907	0.00878146	0.6681062	0	0.00528487	0	0	0.00579836	0	0.01
Dickens_Bleak	0.38560461	0.03966325	0.02574391	0	0.00746108	0.01122307	0	0.03622577	0.13553936	0.0
Eliot_Mill	0	0	0	0.00013472	0.81092871	0	0	0	0	
EBronte_Wuthering	0	0	0	0	0	0.79017129	0	0	0	
Eliot_Middlemarch	0.41050059	0.03073892	0	0	0.0244668	0	0.04544399	0.01505998	0	
Fielding_Joseph	0	0	0.68584529	0	0	0	0	0	0	
ABronte_Tenant	0.07979882	0.49947888	0.05746572	0.01935442	0.00820539	0.03501108	0	0.06832115	0.01676454	0.0
Austen_Emma	0.47033913	0	0	0.00168475	0	0	0	0	0	
Trollope_Prime	0.02513341	0.08664265	0.02245183	0.01082485	0.0059639	0.00583817	0.0103935	0.66317794	0.01937013	0.01
CBronte_Villette	0	0.70215159	0	0.00740248	0	0	0	0	0	
CBronte_Jane	0.17112971	0.50712239	0.0087046	0.00661548	0	0.02615943	0	0	0.01218251	0.01
Richardson_Clarissa	0.09826476	0.06446705	0.45307303	0.0353073	0	0	0.05150161	0.03521831	0.00441265	

H: Topics / Tokens Matrix

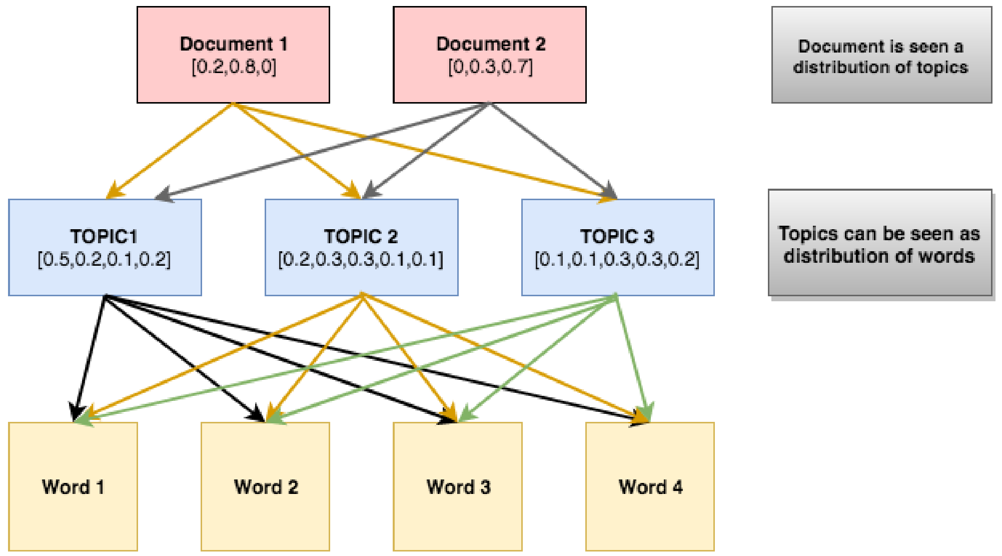


Matrix H: 10 x 64

	adams	allworthy	bounderby	brandon	catherine	cathy	corporal	crawley	darcy	dash
Topic 1	0	0	0	0	0.01987231	0	0	0	0.17684615	
Topic 2	0	0	0	0	0	0	0	0	0	
Topic 3	0.37419098	0.19001803	0	0	0	0	0	0	0	
Topic 4	0	0	0	0.1887578	0.00067853	0	0	0	0.00964817	0.3
Topic 5	0	0	0	0	0	0	0	0	0	
Topic 6	0	0	0	0	0.27943556	0.19547996	0	0	0	
Topic 7	0	0	0	0	0	0	0.14080901	0	0	
Topic 8	0	0	0	0	0	0	0	0	0	
Topic 9	0	0	0	0	0	0	0.00057847	0.27348877	0	
Topic 10	0	0	0.47341063	0	0	0	0	0	0	

Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation



Latent: hidden

Dirichlet: type of probability distribution

Given a document-terms matrix, LDA returns

A ***Document Topic Distribution***

- Each document is represented by a distribution over topics

A Term Topic Distribution

- Each topic is represented by a distribution over all tokens in the vocabulary

A Probabilistic Model

- LDA is a probabilistic topic model that assumes documents are a mixture of topics and that each word in the document is attributable to the document's topics.
- LDA makes a prior assumption that the (documents, topics) and (topics, tokens) mixtures follow Dirichlet probability distributions
- This assumption encourages
    - ***documents to consist of a few topics***
    - ***topics to consist of a few tokens***

Implementing LDA

- Convert documents into bags of words
- Specify the number of topics
- Apply the LDA topic modeling algorithm
- This will output:
    - The distribution of words for each topic
    - The distribution of topics for each document

Evaluation

Evaluating Topic Models

- Automatic Metrics of topic coherence
- Visualising the topic graph
- Human Judgements
    - Do the words associated with each topic form a coherent group ?
    - Can we easily find a label to describe each topic ?
    - Intrusion test

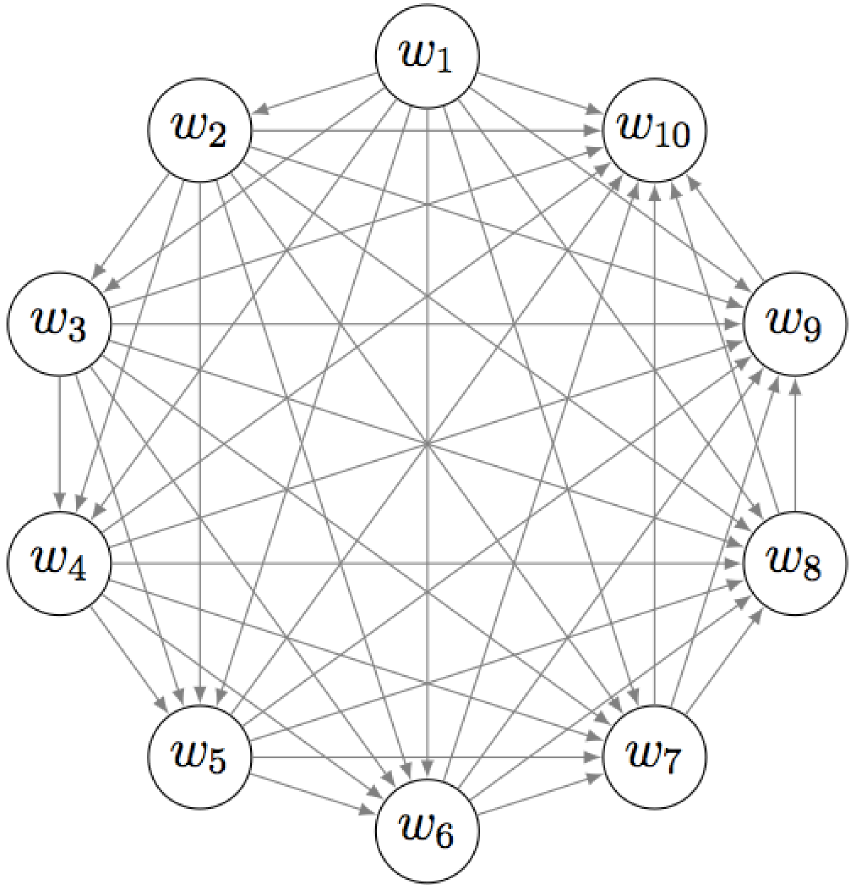
Topic Coherence

- The coherence of a topic is estimated by measuring the degree of semantic similarity between its high scoring words

$$Coherence = \sum score(w_i, w_j)$$

- The words being compared are the ***top words*** associated with the topic

- ***Different metrics*** can be use for similarity



Relevant Tokens for each Topic

The top scoring words of a topic

- are ***specific to this topic***
  - high $p(w \mid t)$ of word w for topic t
- ***distinguish this topic*** from the others
  - non null ***lift value*** $\frac{p(w|t)}{p(w)}$

Relevant Tokens for each Topic

$$relevance(w \mid t) = \lambda . p(w \mid t) + (1 - \lambda) . \frac{p(w \mid t)}{p(w)}$$

The ***lift value***

- is proportional to the ratio of the frequency of a word's occurrence in a specific topic and the frequency of that word in the corpus.
- decreases the score of globally frequent terms
- increases the score of rare terms that occur in a topic
- λ is usually set to 0.3

The coherence of a topic is estimated by measuring the degree of semantic ***similarity*** between its high scoring words (sum of pairwise distributional similarity scores over the set of topic words)

$$Coherence = \sum score(w_i, w_j)$$

UCI Coherence/Similarity Score

$$score_{UCI}(w_i, w_j) = log\frac{p(w_i, w_j)}{p(w_i)p(w_j)}$$

- The UCI Coherence score uses Pointwise Mutual Information (PMI)
- The word probabilities are estimated by counting word co-occurrence frequencies in a sliding window over an external corpus, such as Wikipedia.

UMass Score

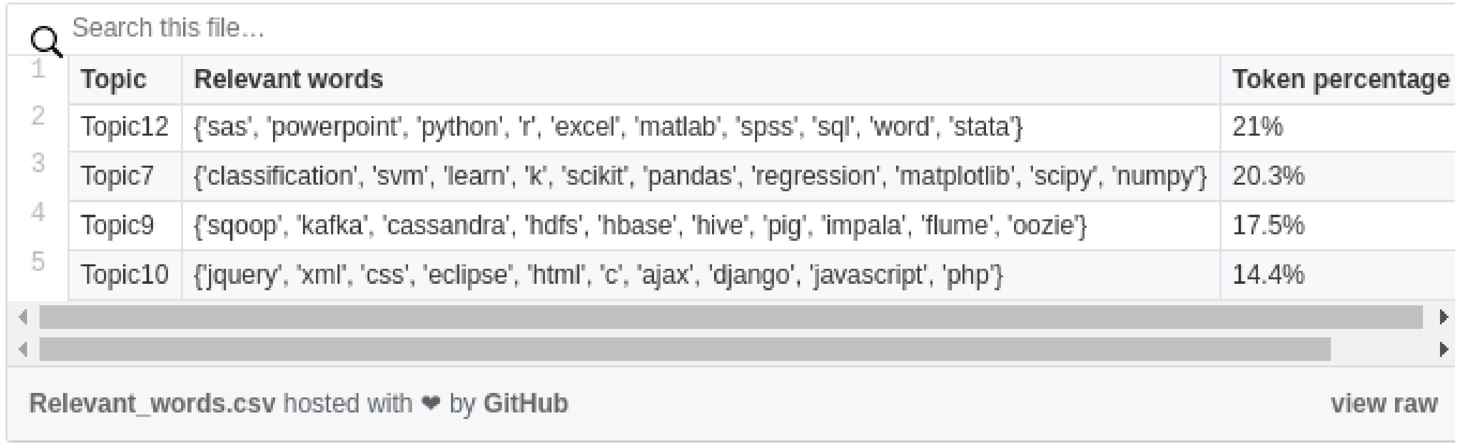
$$scoreUMass(w_i, w_j) = log\frac{D(w_i, w_j)}{D(w_j)}$$

with $D(x, y)$, the number of documents containing words x and y and $D(x)$, the number of documents containing x.

- The UMass coherence score is based on document co-occurrences
- These counts are computed over the original corpus used to train the topic models, rather than an external corpus. This metric is more intrinsic in nature. It attempts to confirm that the models learned data known to be in the corpus.

Human Evaluation

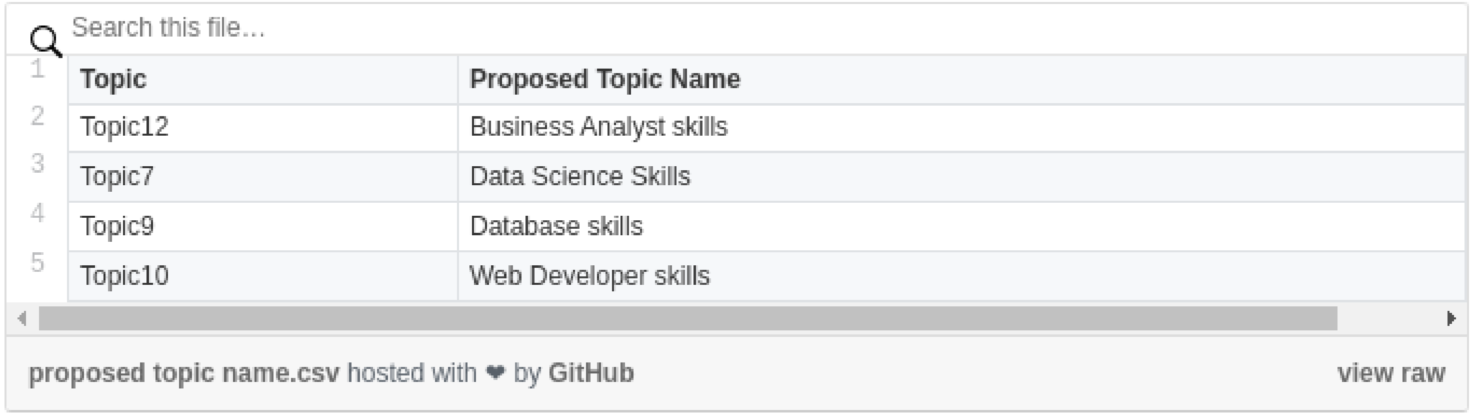
Can we label the topics ?



Search this file…

Topic	Relevant words	Token percentage
Topic12	{'sas', 'powerpoint', 'python', 'r', 'excel', 'matlab', 'spss', 'sql', 'word', 'stata'}	21%
Topic7	{'classification', 'svm', 'learn', 'k', 'scikit', 'pandas', 'regression', 'matplotlib', 'scipy', 'numpy'}	20.3%
Topic9	{'sqoop', 'kafka', 'cassandra', 'hdfs', 'hbase', 'hive', 'pig', 'impala', 'flume', 'oozie'}	17.5%
Topic10	{'jquery', 'xml', 'css', 'eclipse', 'html', 'c', 'ajax', 'django', 'javascript', 'php'}	14.4%

Relevant_words.csv hosted with ❤ by **GitHub** **view raw**



Search this file…

Topic	Proposed Topic Name
Topic12	Business Analyst skills
Topic7	Data Science Skills
Topic9	Database skills
Topic10	Web Developer skills

proposed topic name.csv hosted with ❤ by **GitHub** **view raw**

Word Intrusion

Spot the intruder word

- Select a topic at random
- Choose the 5 most probable words for that topic
  E.g., {dog, cat, horse, pig, cow}
- Add an improbable words for that topic
  E.g., {dog, cat, horse, ***apple***, pig, cow}
- Ask human judges to identify the intruder

If the topics set is coherent, humans will agree on the intruder. Else, they will choose it at random

Chang et al. NIPS 2009

Word Intrusion Example

Coherent topic: the intruder is easy to identify

- {dog, cat, horse, ***apple***, pig, cow}

Incoherent topic: the intruder is hard to identify

- {car, teacher, platypus, agile, blue, Zaire}

Visualising the topic graph

pyLDAvis helps visualise

- Topic prevalence
- Topic similarity
- Words associated with topics

Playing with Lambda

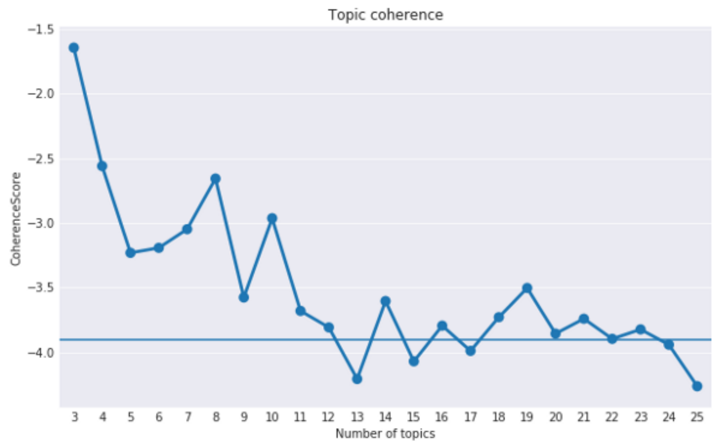
$$relevance(w \mid t) = \lambda.p(w \mid t) + (1 - \lambda).\frac{p(w \mid t)}{p(w)}$$

- $\lambda = 1$, relevance = topic-specific probability (may contain common words)
- $\lambda = 0$, relevance = lift (distinguishing words)
- Modifying the λ value permits visualising the change in relevant terms

PyLDAvis

Online demo

Determining the number of topics



- Plot coherence against number of topics
- Choose the number of topics for which the coherence score plateaus (here around 12-14)

Lab Session

In this lab session, we will apply topic modeling to the film synopsis of movies_500.csv.

The exercises cover the following points:

- Preparing the data
- Training a topic model using LDA
- Inspecting the results using the coherence metrics and visualising the topic graph

Topic Models in Python

Creating the document/tokens matrix

```python
import gensim.corpora as corpora

# Create a Gensim dictionary object from the texts
# "corpus" is a list of lists where each list is the list of tokens contained in a corpus document
dictionary = corpora.Dictionary(corpus)

# Remove extremes  (tokens with low or high scores)
dictionary.filter_extremes(no_below=1, no_above=0.8)

# Create the document/token matrix by creating a BOW vector for each document in your corpus
doc_token_matrix = [dictionary.doc2bow(doc) for doc in corpus]
```

Training an LDA Model

```
# Decide on the number of topics
# chunksize = size of the corpus used at each pass
# passes = number of training iteration over corpus

 lda = models.LdaModel(doc_token_matrix, num_topics=5,
                           id2word=dictionary,
                           update_every=5,
                           chunksize=10000,
                           passes=100)
```

Inspecting an LDA Model

```
# Show the top 20 words associated with each topics together with their probability
# Reminder: A topic is a distribution over words
lda.show_topics(num_words=20)
```

Computing Coherence

```
from gensim.models import CoherenceModel

# Compute Coherence Score
# "corpus" is the document/tokens matrix used as input to GenSim corpora.Dictionary method

coherence_model_lda = CoherenceModel(model=lda_model, texts=corpus, dictionary=dic, coherence='c_v')
coherence_lda = coherence_model_lda.get_coherence()
print('Coherence Score: ', coherence_lda)
```

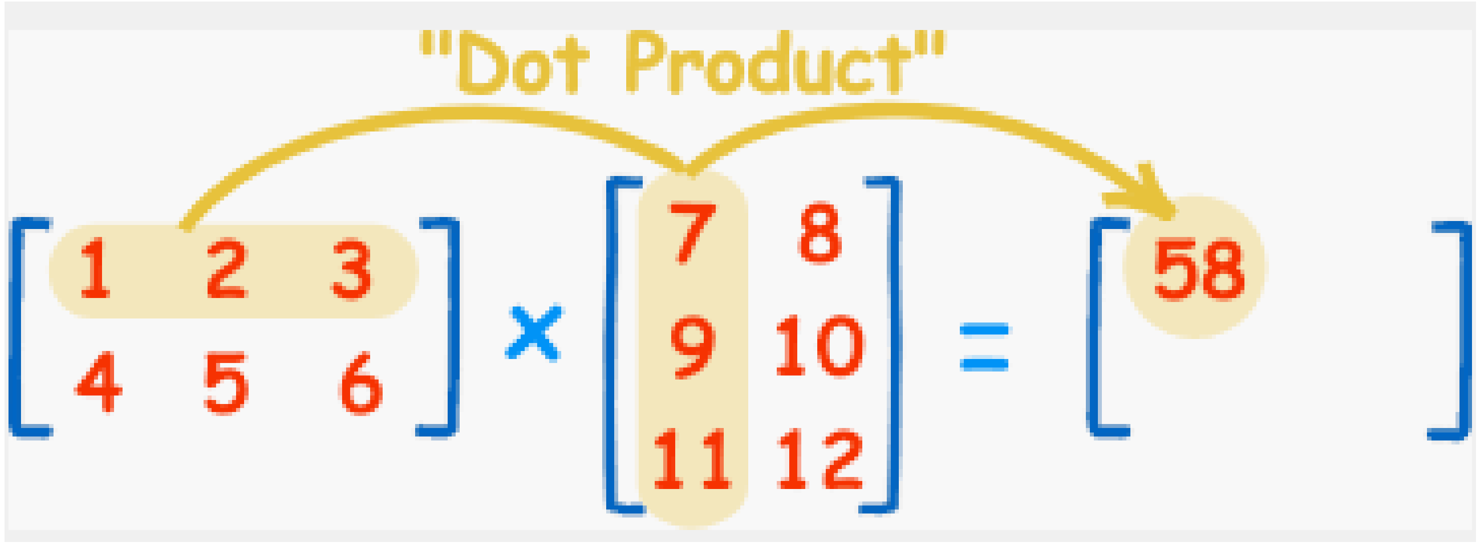
Useful Links

- Topic Modeling with LDA: Blog with code
- LDA to identify topics in a corpus of text synopses : Notebook
- Visualing Topic Models with pyLDAvis: Notebook
- FastAI Notebook on Topic Modeling: SVD, NMF, Truncated SVD
- Randomized-SVD: Blog
- Online Matrix Multiplication Tool
- Book Chapter on Matrix Decomposition
- Blog and interactive visualisation using LDA topic modeling

Linear Algebra: Matrix Multiplication

Online Matrix Multiplication Tool

- $C_{m,n} = A_{m,k} \times B_{k,n}$
- Dot product of rows and columns
  E.g., $(1 \times 7) + (2 \times 9) + (3 \times 11) = 58$



Matrix Multiplication

- $C_{m,n} = A_{m,k} \times B_{k,n}$
- Each column of C is a linear combination of rows in B where the coeffficients come from the corresponding column in A
- Typically, in Machine Learning for NLP, features and weights are stored in matrices: A are the weights and B the features

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix}, \mathbf{B} = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{pmatrix}$$

$$\mathbf{AB} = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} & a_{11}b_{13} + a_{12}b_{23} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} & a_{21}b_{13} + a_{22}b_{23} \\ a_{31}b_{11} + a_{32}b_{21} & a_{31}b_{12} + a_{32}b_{22} & a_{31}b_{13} + a_{32}b_{23} \end{pmatrix}$$

$$\mathbf{BA} = \begin{pmatrix} b_{11}a_{11} + b_{12}a_{21} + b_{13}a_{31} & b_{11}a_{12} + b_{12}a_{22} + b_{13}a_{32} \\ b_{21}a_{11} + b_{22}a_{21} + b_{23}a_{31} & b_{21}a_{12} + b_{22}a_{22} + b_{23}a_{32} \end{pmatrix}$$