Topic Modeling

Section Goals

- Understand Topic Modeling
- Learn Latent Dirichlet Allocation
- Implement LDA
- Understand Non-Negative Matrix Factorization
- Implement NMF
- Apply LDA and NMF with a project

What is Topic Modelling

- Before we learn about methods such as Latent Dirichlet Allocation or Non-negative Matrix Factorization, let’s understand the general concept of Topic Modeling and why it’s important!

Topic Modelling is unsupervised

- Topic Modeling allows for us to efficiently analyze large volumes of text by clustering documents into topics.
- A large amount of text data is **unlabeled** meaning we won't be able to apply our previous supervised learning approaches to create machine learning models for the data!

Grouping by Topics

- If we have **unlabeled** data, then we can attempt to “discover” labels.
- In the case of text data, this means attempting to discover clusters of documents, grouped together by topic.

Cluster Similar Ideas

- A very important idea to keep in mind here is that we don't know the "correct" topic or "right answer"!
- All we know is that the documents clustered together share similar topic ideas.
- It is up to the user to identify what these topics represent.

LDA for Documents

We will begin by examining how Latent Dirichlet Allocation can attempt to discover topics for a corpus of documents!

Latent Dirichlet Allocation

With Python - Part One

LDA History

- Johann Peter Gustav Lejeune Dirichlet was a German mathmatician in the 1800s who contributed widely to the field of modern mathematics.
- There is a probability distribution named after him “Dirichlet Distribution”.
- In 2003 LDA was first published as a graphical model for topic discovery in *Journal of Machine Learning Research* by David Blei, Andrew Ng and Michael I. Jordan.

LDA Overview

- Let’s get a high level overview of how LDA works for topic modelling.
- I encourage you to also take a look at the original publication paper!

Assumptions of LDA for Topic Modeling

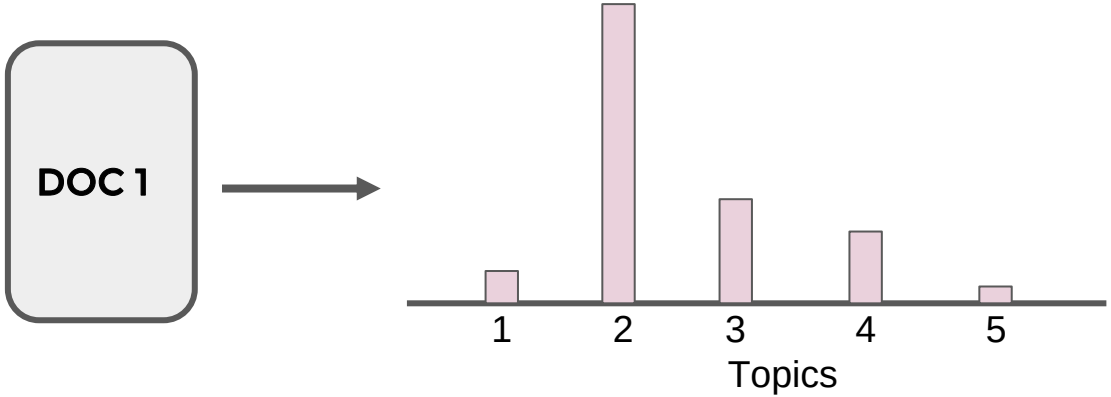
- Documents with similar topics use similar groups of words
- Latent topics can then be found by searching for groups of words that frequently occur together in documents across the corpus.

Assumptions of LDA for Topic Modeling

- Documents are probability distributions over latent topics.
- Topics themselves are probability distributions over words.

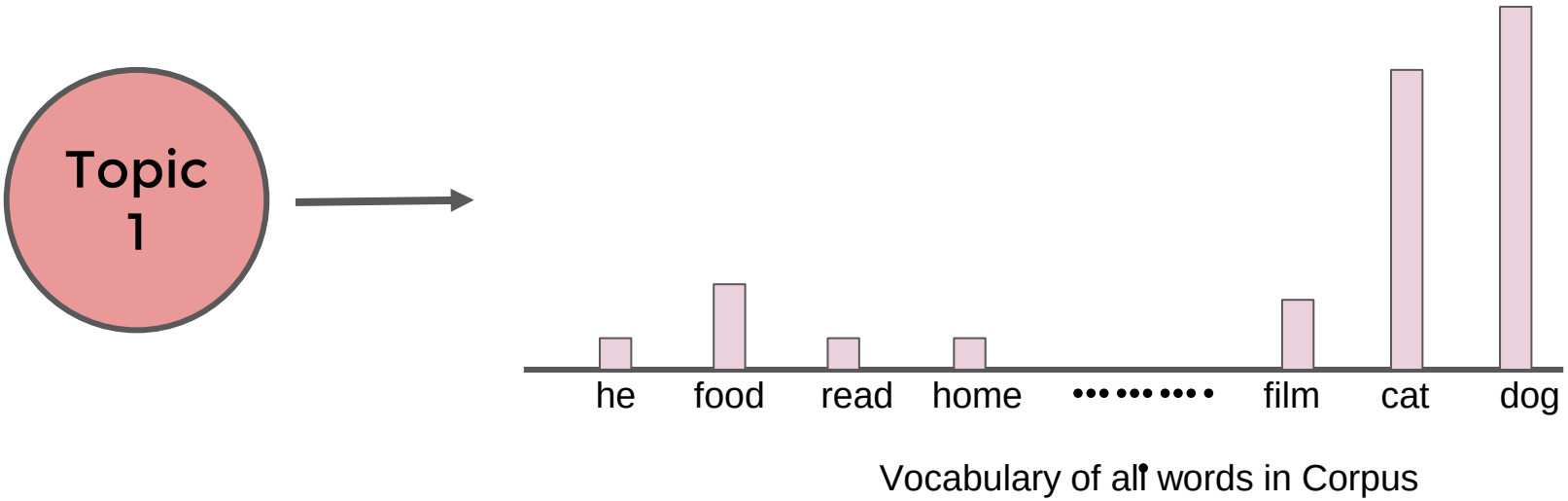
Doc -> Topics

Documents are probability distributions over latent topics.



Topics -> Words

Topics themselves are probability distributions over words.

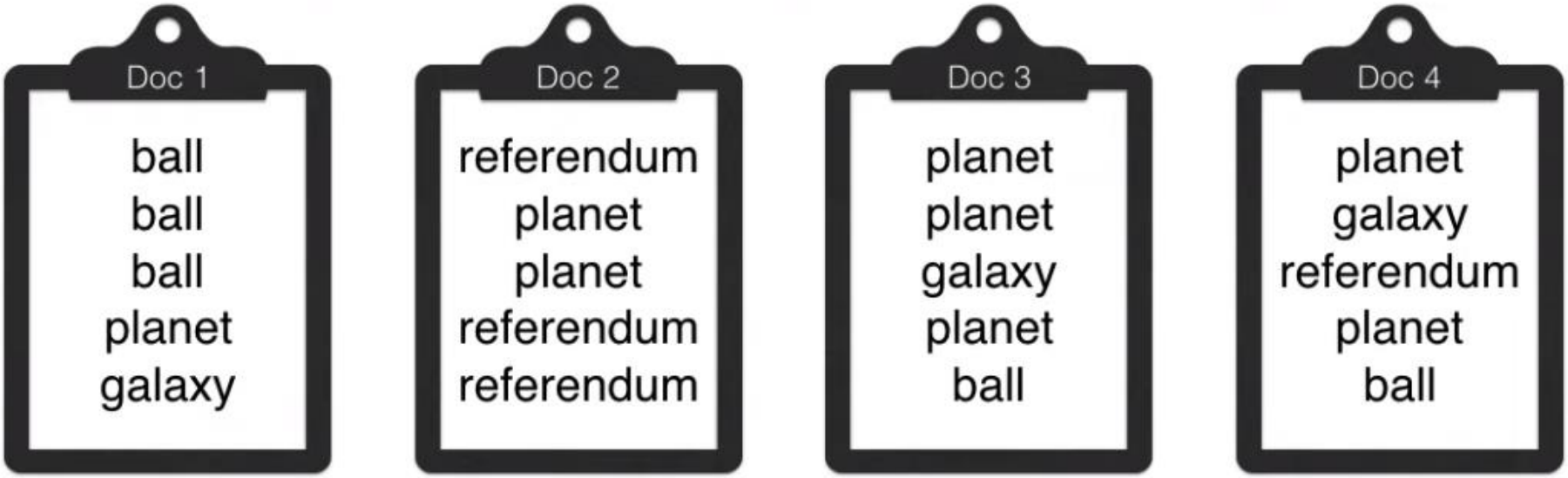


LDA

- **LDA represents documents as mixtures of topics that spit out words with certain probabilities.**
    - Note that the probabilities are from drichlet distribution.

- Our goals is to find the best parameters that spits out words and topics which matches the current document corpus.

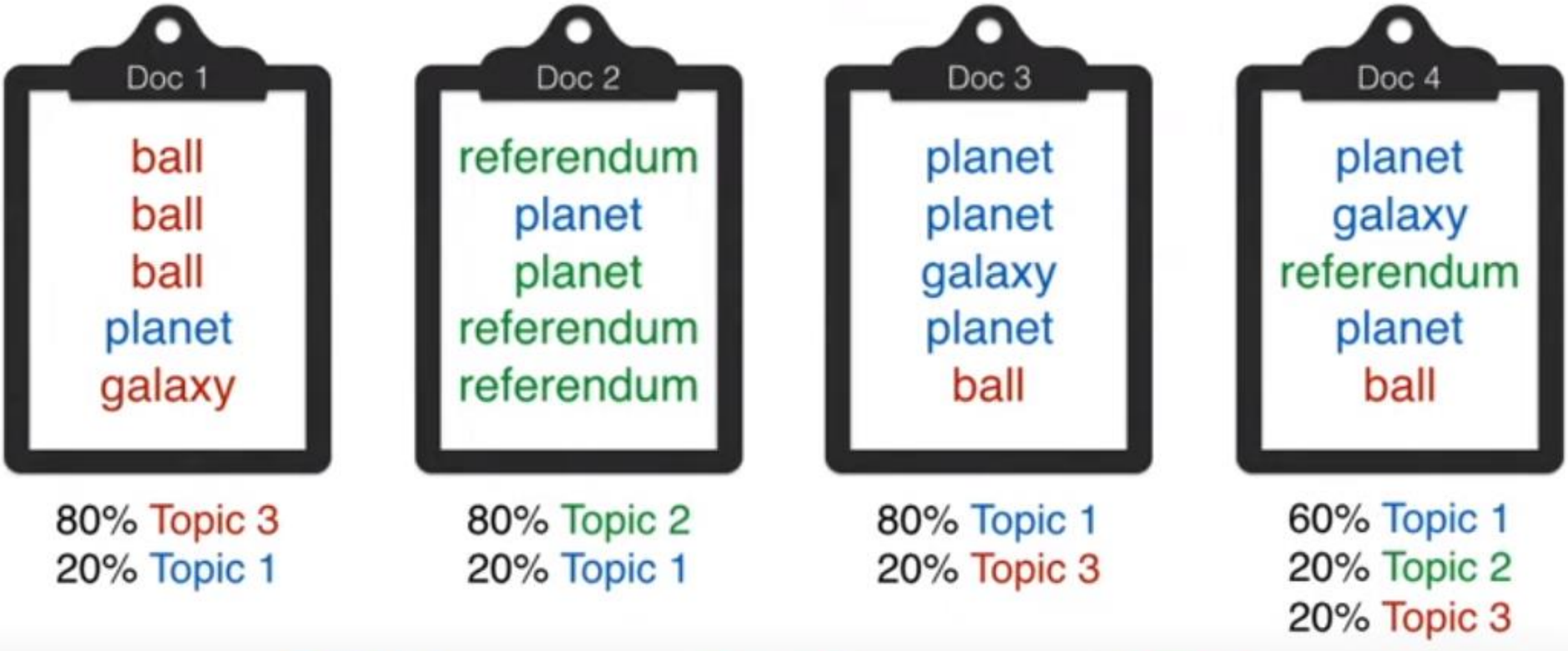
Documents vs Topics



Sports

Science Politics Sports

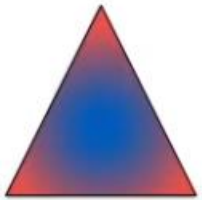
After LDA


Doc 1
ball
ball
ball
planet
galaxy
Doc 2
referendum
planet
planet
referendum
referendum
Doc 3
planet
planet
galaxy
planet
ball
Doc 4
planet
galaxy
referendum
planet
ball
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
Topic 1
Topic 2
Topic 3

Words are as monochromatic as possible

Probability of a document

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



LDA Principle

It assumes that documents are produced in the following fashion:

- Decide on the number of words N the document will have.
- Choose a topic mixture for the document (according to a Dirichlet distribution over a fixed set of M topics).
- e.g. 60% business, 20% politics, 10% food

Generate each word in the document

First picking a topic according to the multinomial distribution that you sampled previously (60% business, 20% politics, 10% food)

Using the topic, generate the word itself (according to the topic's multinomial distribution).

For example, if we selected the food topic, we might generate the word "apple" with 60% probability, "home" with 30% probability, and so on.

Backtrack from the document

Assuming this generative model for a collection of documents, LDA then tries to backtrack from the documents to find a set of topics that are likely to have generated the collection.

Imagine we have a set of documents.

We've chosen some fixed number of M topics to discover, and want to use LDA to learn the topic representation of each document and the words associated to each topic.

Go through each document

- Randomly assign each word in the document to one of the M topics at a time. Aka. GIBBS Sampling
- This random assignment already gives you both topic representations of all the documents and word distributions of all the topics (note, these initial random topics won't make sense).

Start Random

Goal: Color each word with blue, green, red

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

Improve Topics

Now we iterate over every word in every document to improve these topics.

For every word in every document and for each topic **t**, calculate:

- p(topic **t** | document **d**) = the proportion of words in document **d** that are currently assigned to topic **t**

p(topic **t** | document **d**) => 2, 0, 2

Topic 1 Topic 2 Topic 3

How much is Topic 1 in Doc 1? How much is Topic 2 in Doc 1? How much is Topic 3 in Doc 1?

p(word w | topic t)

- Calculate the probability that topic t generated word w

Topic 1

How much is Topic 1 in Doc 1?

2

How much is ‘ball’ in Topic 1?

0

Topic 2

How much is Topic 2 in Doc 1?

0

How much is ‘ball’ in Topic 2?

1

Topic 3

How much is Topic 3 in Doc 1?

2

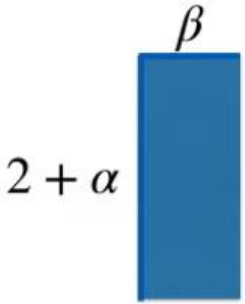
How much is ‘ball’ in Topic 3?

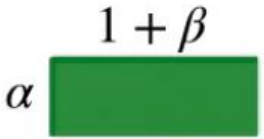
3

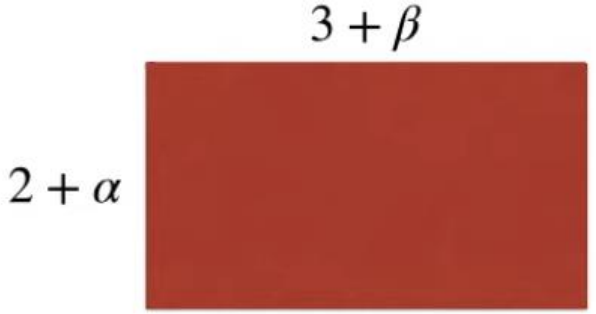
Reassign a new topic

- Reassign w a new topic, where we choose topic t with probability **p(topic t | document d) * p(word w | topic t)**
- **That is 2 x 0 vs 0 x 1 vs 2 x 3**
- **To eliminate multiply by «0» add Alpha and Beta parameters from Drichlet Distribution**

Likely the ball will be RED but others are possible also.







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?

$3+\beta$

Iterate until steady state

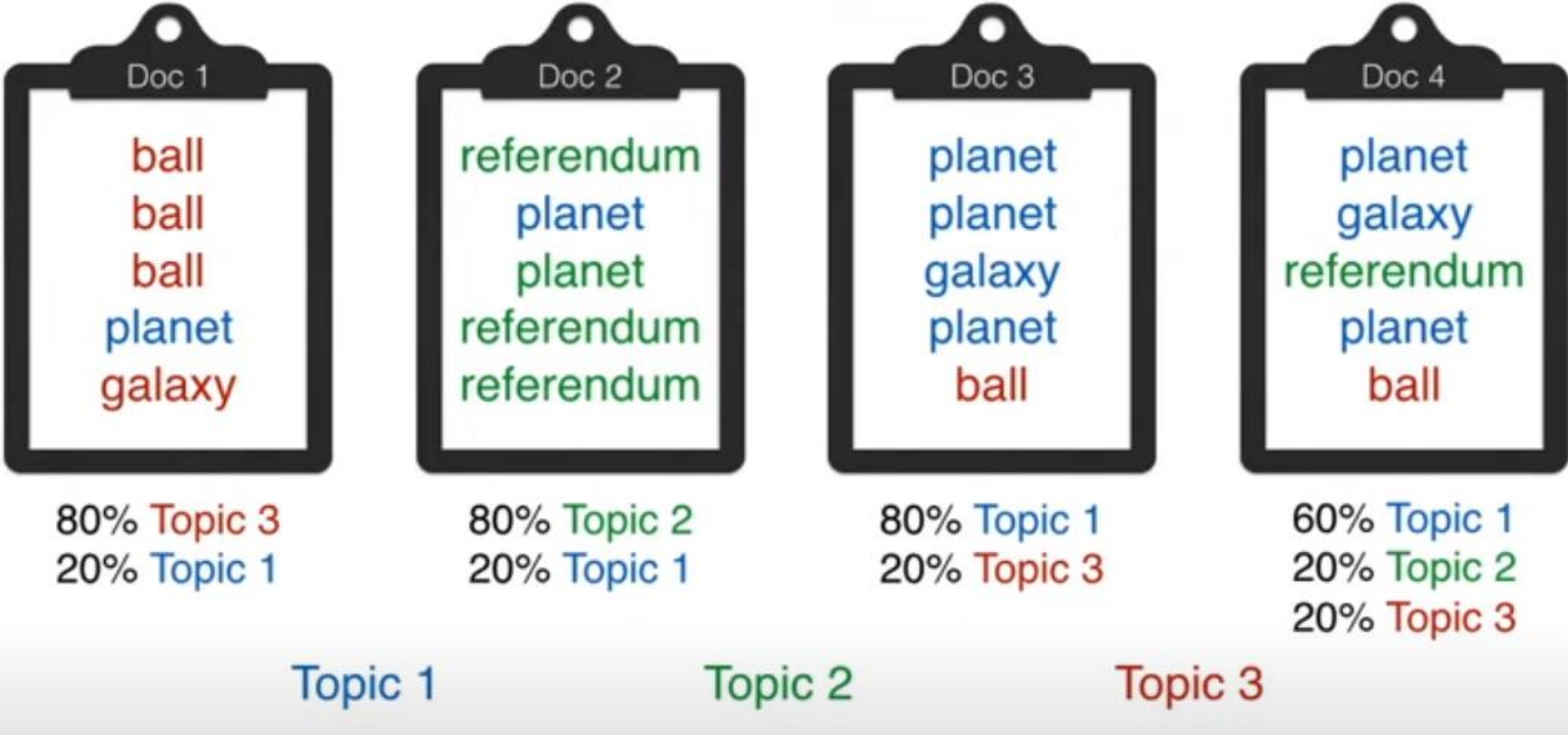
After repeating the previous step, many times, we eventually reach a steady state where the assignments are acceptable.

- At the end, each document assigned to a topic.
- We also can search for the words that have the highest probability of being assigned to a topic.

Output of LDA

- We end up with an output such as:
  - Document assigned to Topic #4
  - Most common words (highest probability) for Topic #4:
    - [‘cat’,’vet’,’birds’,’dog’,...,’food’,’home’]
  - It is up to the user to interpret these topics.

After LDA


Doc 1
ball
ball
ball
planet
galaxy
Doc 2
referendum
planet
planet
referendum
referendum
Doc 3
planet
planet
galaxy
planet
ball
Doc 4
planet
galaxy
referendum
planet
ball
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
Topic 1
Topic 2
Topic 3

User Involvement

- Two important notes:
  - The user must decide on the amount of topics present in the document.
  - The user must interpret what the topics are.

Latent Dirichlet Allocation with Python

PART TWO

Non-Negative Matrix Factorization

Document Term Matrix

Count Vectorizer

Terms

Documents

	Planet	Election	Soccer	World	Finance
Document - 1	5	1	1	4	3
Document - 2	2	5	3	4	3
Document - 3	1	4	1	3	5
Document - 4	1	2	5	3	4

Given Terms can you come up with Topics for Each Document ?

DTM ⟹ TF-IDF ⟹ Topics

Terms

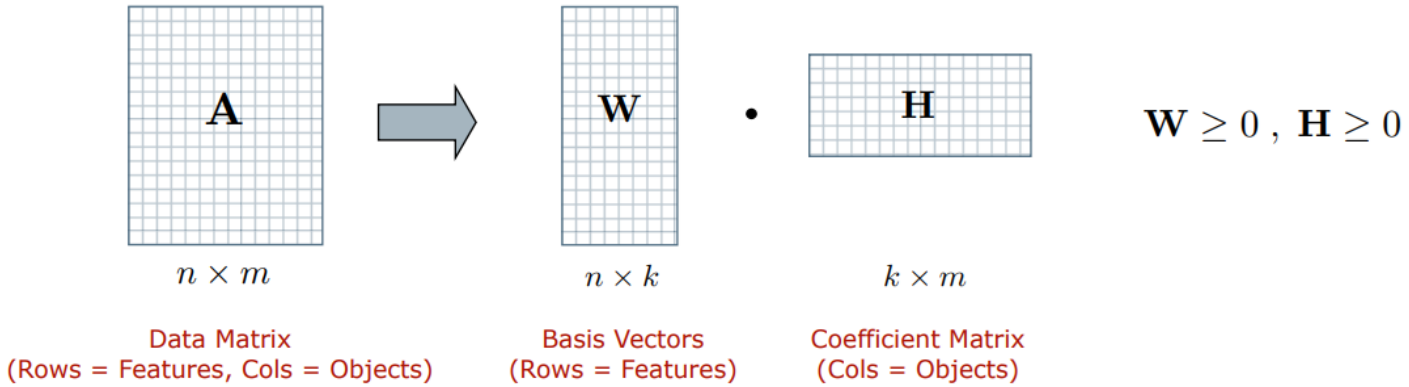
Documents

	Planet	Election	Soccer	World	Finance
Science	0.35	0.11	0.11	0.26	0.21
Politics	0.74	…			
Economy					
Sports					

NnMF is an unsupervised algorithm that simultaneously performs clustering and dimensionality reduction

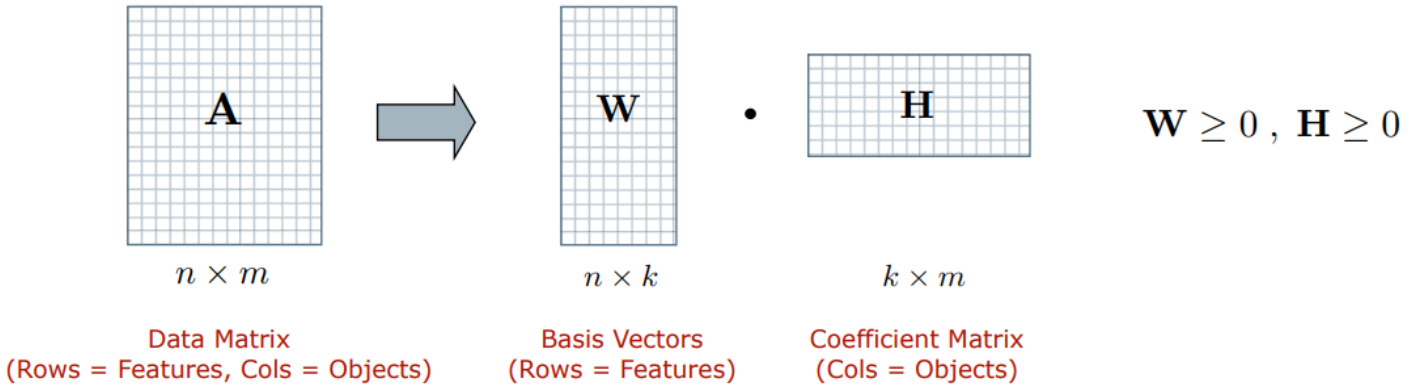
NnMF

- Given a non-negative matrix A (Transpose of DTM), find k-dimension (Topics) approximation in terms of non-negative factors W and H



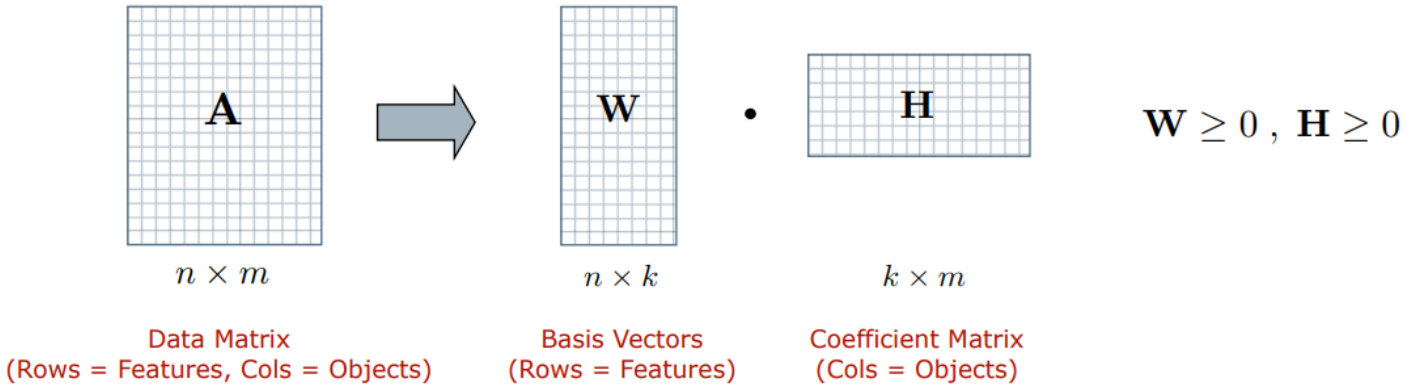
NnMF

Approximate each object (i.e. row of A) by a linear combination of k reduced dimensions or “basis vectors” in W.



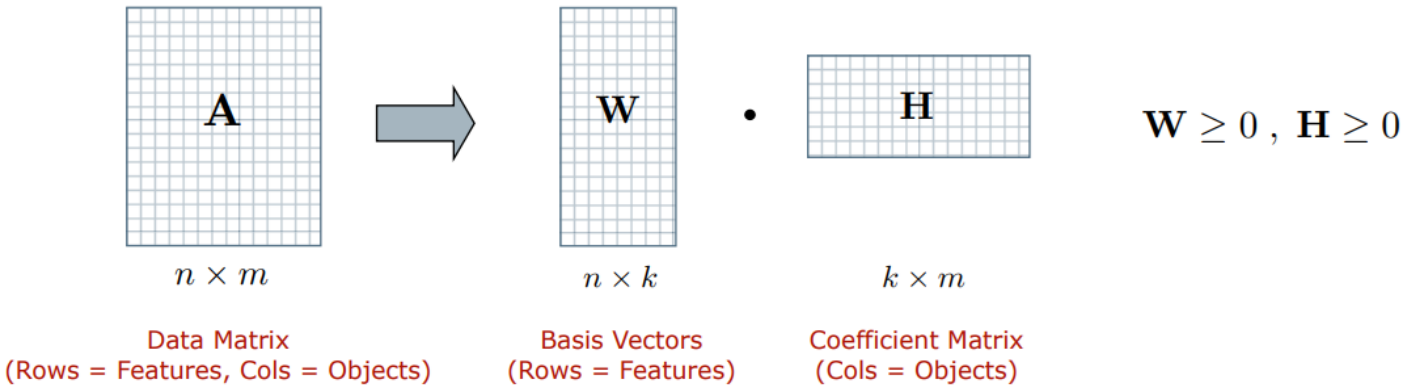
NnMF

Each basis vector can be interpreted as a cluster.
The memberships of objects in these clusters encoded by H.



NnMF Input

Given: A and k Topics
Find: W and H



NnMF Objective Function

- Assign initial values for W and H (e.g. random matrices)
- Measure error between A and the approximation WH

$$\frac{1}{2}||\mathbf{A}-\mathbf{WH}||_{\mathrm{F}}^2=\sum_{i=1}^{n}\sum_{j=1}^{m}(A_{ij}-(WH)_{ij})^2$$

NmMF Optimization

Minimise the objective function (Error) by refining W and H.
Common approach is to iterate between two multiplicative update rules until convergence

1. Update **H**

$$H_{cj} \leftarrow H_{cj} \frac{(W\mathbf{A})_{cj}}{(W\mathbf{WH})_{cj}}$$

2. Update **W**

$$W_{ic} \leftarrow W_{ic} \frac{(\mathbf{A}H)_{ic}}{(\mathbf{WH}H)_{ic}}$$

NnMF Steps

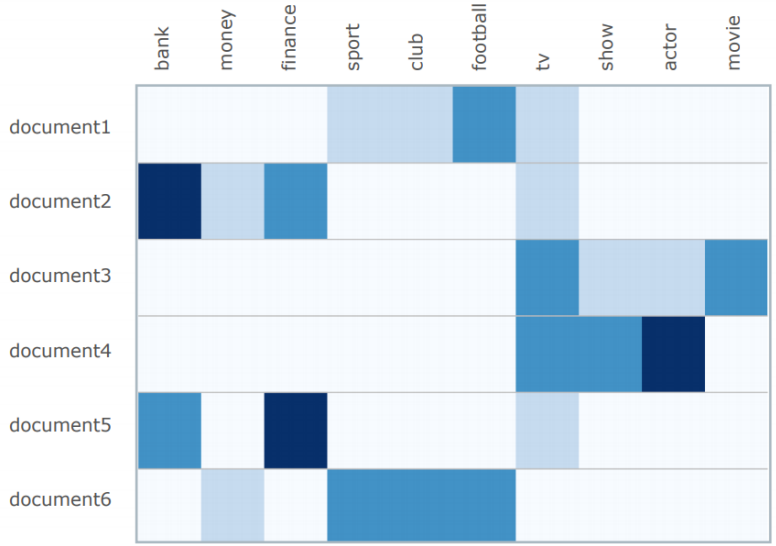
1. Construct vector space model for documents (after stopword filtering), resulting in a term-document matrix A.
   - Apply TF-IDF term weight normalization to A
   - Normalize TF-IDF vectors to unit length.
2. Initialise factors using NND-SVD (Nonnegative Double Singular Value Decomposition) on A.
3. Apply Projected Gradient NMF to A.

NnMF Vectors

- Basis vectors: the topics (clusters) in the data.
- Coefficient matrix: the membership weights for documents relative to each topic (cluster).

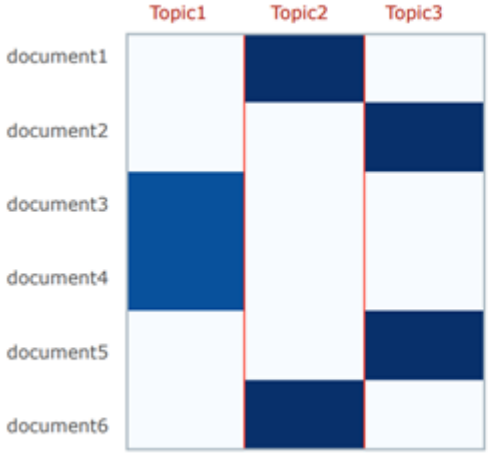
Term Matrix with TF-IDF

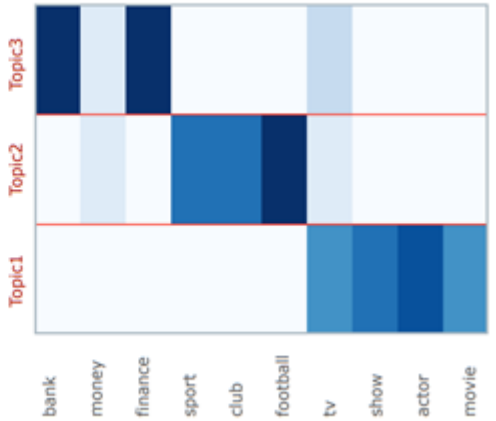
Create a document term matrix with TF-IDF Vectorization.



Natural Language Processing Bootcamp

Resulting W and H





How to obtain W and H

	M1	M2	M3	M4	M5
 Comedy	3	1	1	3	1
 Action	1	2	4	1	3

	 Comedy	 Action
A	1	0
B	0	1
C	1	0
D	1	1

	M1	M2	M3	M4	M5
A	3	1	1	3	1
B	1	2	4	1	3
C	3	1	1	3	1
D	4	3	5	4	4

Gradient Descent – Random First Entry

	M1	M2	M3	M4	M5
F1	1.2	3.1	0.3	2.5	0.2
F2	2.4	1.5	4.4	0.4	1.1

$$1.2 \times 0.2 + 2.4 \times 0.5 = 1.44$$

	F1	F2
 A	0.2	0.5
 B	0.3	0.4
 C	0.7	0.8
 D	0.4	0.5

	M1	M2	M3	M4	M5
	**1.44**	**1.37**	**2.26**	**0.7**	**0.59**
	**1.32**	**1.53**	**1.85**	**0.91**	**0.5**
	**2.76**	**3.37**	**3.73**	**2.07**	**1.02**
	**1.68**	**1.99**	**2.32**	**1.2**	**0.63**

Update W and H Matrix so we get closer to target

	M1	M2	M3	M4	M5
F1	1.4	3.1	0.3	2.5	0.2
F2	2.5	1.5	4.4	0.4	1.1

$1.4 \times 0.3 + 2.5 \times 0.6 = 1.92$

	F1	F2
A	0.3	0.6
B	0.3	0.4
C	0.7	0.8
D	0.4	0.5

	M1	M2	M3	M4	M5
	1.92				

	M1	M2	M3	M4	M5
	3	1	1	3	1
	1	2	4	1	3
	3	1	1	3	1
	4	3	5	4	4

Update W and H Matrix for every entry

	M1	M2	M3	M4	M5
F1	1.4	3.1	0.3	2.5	0.2
F2	2.5	1.5	4.4	0.4	1.1

$3.1 \times 0.3 + 1.5 \times 0.6 = 1.83$

	F1	F2
A	0.3	0.6
B	0.3	0.4
C	0.7	0.8
D	0.4	0.5

	M1	M2	M3	M4	M5
	1.92	1.83			

	M1	M2	M3	M4	M5
	3	1	1	3	1
	1	2	4	1	3
	3	1	1	3	1
	4	3	5	4	4

How much move up or down. Decide by Gradient Descent

	M1	M2	M3	M4	M5
F1	1.2	3.1	0.3	2.5	0.2
F2	2.4	1.5	4.4	0.4	1.1

Derivative

$$\text{Error} = (3 - 1.44)^2 + (1 - 1.37)^2 + \dots$$

	F1	F2
A	0.2	0.5
B	0.3	0.4
C	0.7	0.8
D	0.4	0.5

	M1	M2	M3	M4	M5
A	1.44	1.37			
B					
C					
D					

	M1	M2	M3	M4	M5
A	3	1	1	3	1
B	1	2	4	1	3
C	3	1	1	3	1
D	4	3	5	4	4

Example with NnMF

- Just like LDA, we will need to select the number of expected topics beforehand (the value of **k**)!
- Also just like with LDA, we will have to interpret the topics based off the coefficient values of the words per topic.

Example with NnMF

- Luckily, due to Scikit-Learn's uniform syntax, switching out LDA for NMF is very simple.
- Let's repeat our analysis of the NPR Article data set and discover topics with NMF.

Non-Negative Matrix Factorization with Python