

# Topic Modeling or Extraction

- The Importance of Words in Documents
  - Problem: categorizing documents (sequences of words) by their **topic**.
  - Topics are identified by finding the **special words** that characterize documents about that topic.
  - e.g. soccer -> many occurrences words like “penalti” “arbitro,” “goool”, “FIFA” etc.

# Topic Modeling or Extraction

- Thus, classification often starts by looking at documents, and finding the significant words in those documents.
- Our first guess might be that the words appearing most frequently in a document are the most significant.
- However, that intuition is exactly opposite of the truth.
- The most frequent words will most surely be the common words such as “the” or “and,” which help build ideas but do not carry any significance themselves (**topic noise**).
- The several hundred most common words in English (Spanish similar, called **stop words**) are often removed from documents before any attempt to classify them.

○

# Topic Modeling or Extraction

- The indicators of the topic are *relatively rare words*.
- Not all rare words are equally useful as indicators (e.g. “notwithstanding” ,“albeit”)
- “chukker” (polo), “hadron”, “axolotl” is probably equally rare, but tips us off.
- The formal measure of how concentrated into relatively few documents are the occurrences of a given word is called TF.IDF (Term Frequency times Inverse Document Frequency).
- If we were to feed the direct count data directly to a classifier those very frequent terms would **shadow the frequencies** of rarer yet more interesting terms.
- In order to re-weight the **count features** into floating point values suitable for usage by a classifier we use tf-idf

# Topic Modeling or Extraction

- **Term Frequency (tf):** gives us the frequency of the word in each document in the corpus. It is the ratio of number of times the word appears in a document compared to the total number of words in that document. It increases as the number of occurrences of that word within the document increases. Each document has its own tf (a relative frequency)

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}}$$

- Define  $f_{ij}$  to be the frequency of word  $i$  in document  $j$ .

# Topic Modeling or Extraction

- **Inverse Data Frequency (idf):** used to calculate the weight of rare words across all documents in the corpus. The words that occur rarely in the corpus have a high IDF score. It is given by the equation below (N number of docs in the corpus).

$$idf(w) = \log\left(\frac{N}{df_t}\right)$$

- Combining these two we come up with the TF-IDF score (w) for a word in a document in the corpus. It is the product of tf and idf:

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{ij}$  = number of occurrences of  $i$  in  $j$   
 $df_i$  = number of documents containing  $i$   
 $N$  = total number of documents

# Example

- Corpus:
  - Sentence A : The car is driven on the road.
  - Sentence B: The truck is driven on the highway.

Word	TF		IDF	TF*IDF	
	A	B		A	B
The	1/7	1/7	$\log(2/2) = 0$	0	0
Car	1/7	0	$\log(2/1) = 0.3$	0.043	0
Truck	0	1/7	$\log(2/1) = 0.3$	0	0.043
Is	1/7	1/7	$\log(2/2) = 0$	0	0
Driven	1/7	1/7	$\log(2/2) = 0$	0	0
On	1/7	1/7	$\log(2/2) = 0$	0	0
The	1/7	1/7	$\log(2/2) = 0$	0	0
Road	1/7	0	$\log(2/1) = 0.3$	0.043	0
Highway	0	1/7	$\log(2/1) = 0.3$	0	0.043

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
tfidf_vectorizer = TfidfVectorizer()  
tfidf = tfidf_vectorizer.fit_transform(A,B)
```

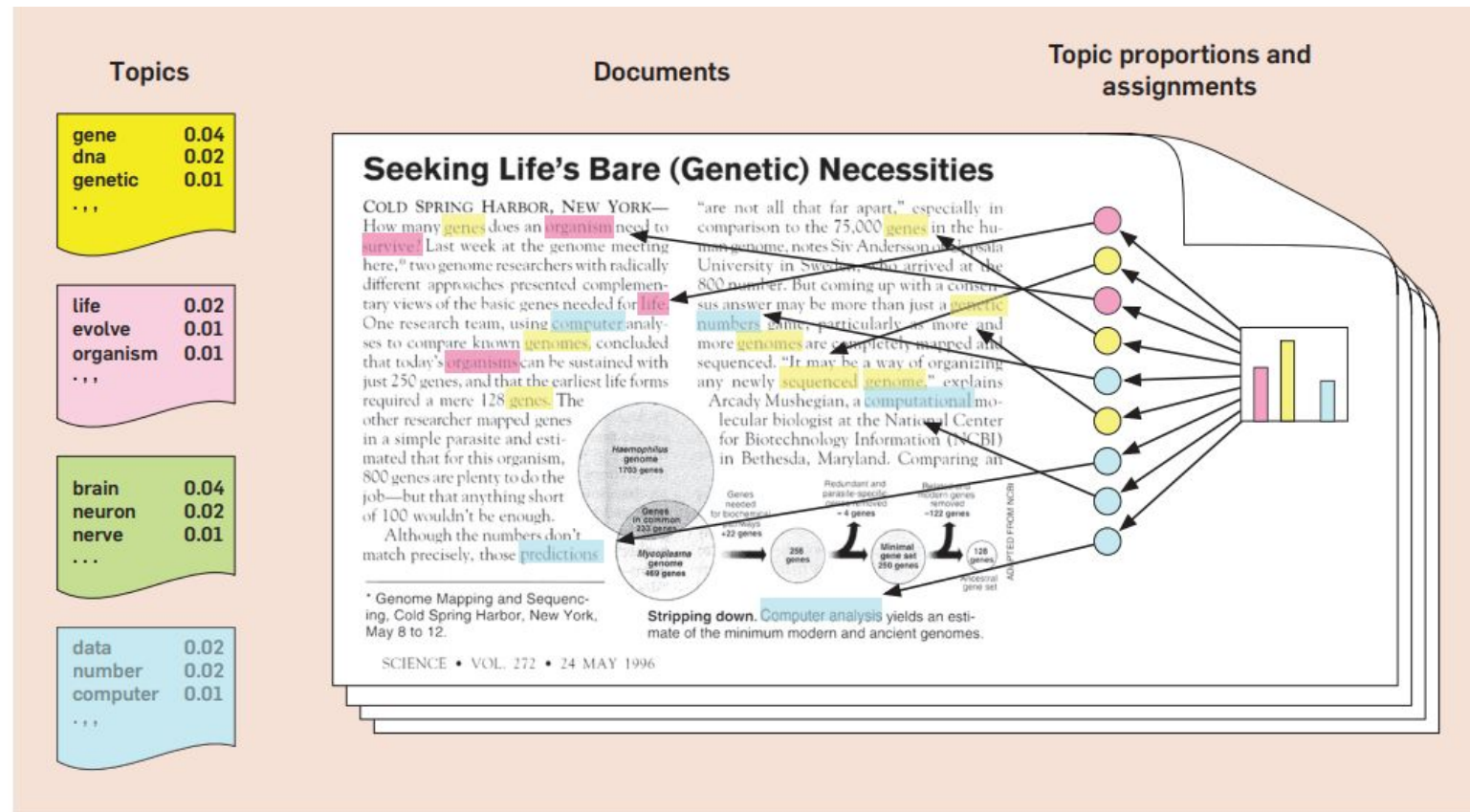
Español

# LDA

- **Latent Dirichlet Allocation (LDA):** modelo probabilístico para inferir una distribución de temas
  - latent (escondido), Dirichlet (distribución de palabras), Allocation (asignación a temas)
  - Es un tipo de clustering (No supervisado): documentos->temas, temas->palabras
  - Objetivo: Dada una bolsa de palabras (documento), determina los temas presentes.



8. David M. Blei , “[Probabilistic Topic Models](#)” . Communications of the ACM, April 2012, Vol. 55 No. 4, Pages 77-84 10.1145/2133806.2133826



## ¿Porque Probabilístico?

- Para asignar palabras a clusters necesita dos probabilidades:
  - **$P(\text{palabra} \mid \text{tema})$**  : una palabra en un tema
  - **$P(\text{tema} \mid \text{documento})$**  : un tema en un documento
- Asignación inicial al azar
- Después procedimiento iterativo para cada palabra del documento para asignarlas a temas, hasta alcanzar convergencia (actualizar probabilidades i.e. Bayesiano)
- Un documento es una distribución de temas
- Un tema es una distribución de palabras que pertenecen a un vocabulario
- Dataset->Corpus: colección de documentos

# Ejemplo

Document 1		Document 2		Document 3	
Eat	A	Cat	B	Cat	B
Fish	A	Dog	B	Eat	A
Vegetables	A	Pet	B	Fish	?
Fish	A	Pet	B	Cat	B
Eat	A	Fish	B	Fish	A

- 3 documentos, dos temas, comida (A) y mascotas (B)
- Asignación de Fish en doc3, ¿a que tema pertenece?
- Doc 1 es casi sobre comida, doc2 sobre mascotas y doc3 mezclado más o menos igual.
- Calcula la probabilidad de que la palabra aparezca en los tres temas i.e.
- $PA(\text{'Fish'} \mid \text{tema A})$  ,  $PB(\text{'Fish'} \mid \text{tema B})$  (*mas frecuencia de Fish en A*)
- Ahora calcula la probabilidad de que el tema aparezca en el documento 3 en cuestión..
- $P(\text{tema A} \mid \text{doc3}) = P(\text{tema B} \mid \text{doc3})$  (*misma frecuencia ambos temas*)
- Como  $PA > PB$ , Concluimos que Fish en doc3 está en el tema A.
- Iterar sobre todas las palabras de todos los documentos, y esto varias veces (o pases)...

# Aprendizaje en LDA - Document-Term matrix

- Escogemos un K número de temas (clusters), N documentos, M vocabulario
- Asignamos palabras a temas en forma aleatoria, construimos:

	W1	W2	W3	<u>Wn</u>
D1	0	2	1	3
D2	1	4	0	0
D3	0	2	3	1
<u>Dn</u>	1	1	3	0

	K1	K2	K3	K
D1	1	0	0	1
D2	1	1	0	0
D3	1	0	0	1
<u>Dn</u>	1	0	1	0

	W1	W2	W3	<u>Wm</u>
K1	0	1	1	1
K2	1	1	1	0
K3	1	0	0	1
K	1	1	0	0

- Para mejorar las distribuciones (i.e. inferir o aprender de los datos/documentos), iteramos:
- Para cada W en doc:
- Tema 1 a K:
  - ¿Cuántas palabras en el doc ya pertenecen al tema 1?  $P1(\text{tema 1} | \text{doc d})$
  - ¿Con qué frecuencia W aparece el tema 1 en todos los docs?  $P2(\text{word W} | \text{tema 1})$
  - $P1 * P2$  = Probabilidad de que W vino del tema 1
  - Si  $(P1 * P2)_{\text{tema 2}} > (P1 * P2)_{\text{tema 1}}$  ----> Cambio a tema 2
  - Continua hasta iteraciones deseadas o estado estacionario

# Implementación

1. Implementación de un código que mina temas en el dataset “all the news”
2. **articles1.csv, ~50k líneas:** Artículos publicados en diversos medios de comunicación de habla inglesa.
3. Este código es relativamente sencillo,  $K=10$  temas,
  - a. Un solo pase, aprox 2 horas para 50k líneas de articles1.csv
4. Corpus (articles1.csv) es bajado a mano
5. Selección de las columnas a usar (headers)
6. Limpieza del corpus
  - a. líneas defectuosas,
  - b. **stop\_words** i.e. palabras irrelevantes e.g. conjunciones: “of”, “or”, etc.
7. **gensim**, *Topic modeling* package para construir el modelo LDA y la matriz documentos-temas.
  - a. Diccionario: id-strings *mappings* , doc2bow (bag of words) format, etc. Escalable.

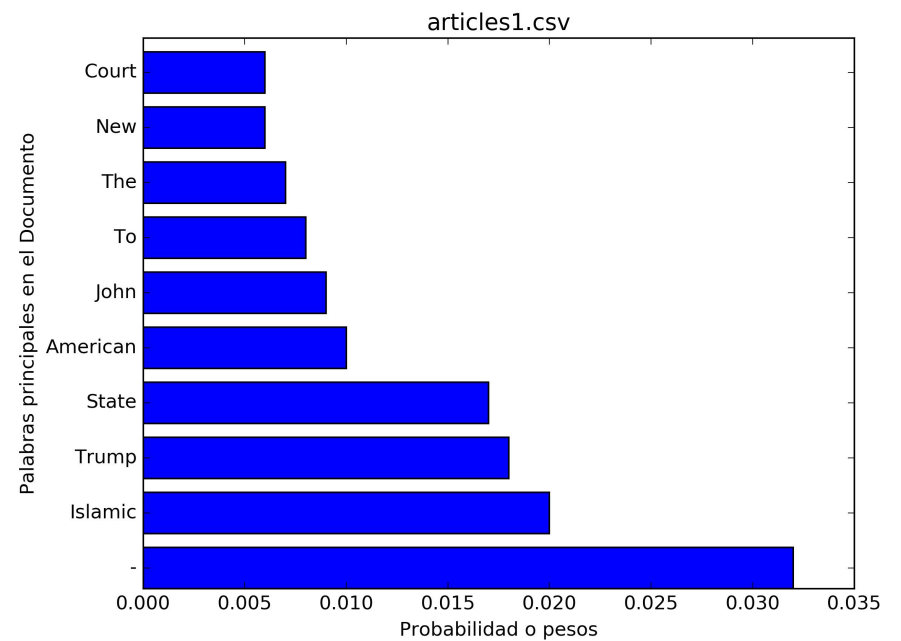
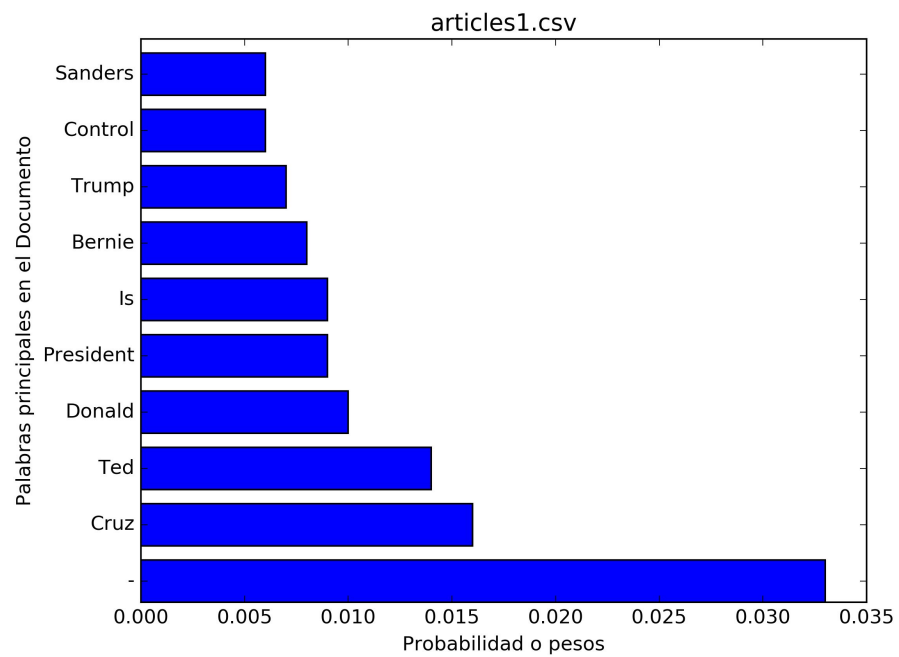
# Resultados - Temas (topic clusters) inferidos

```
benjamin@higgs:~/topic$ python topic_modeling.py
```

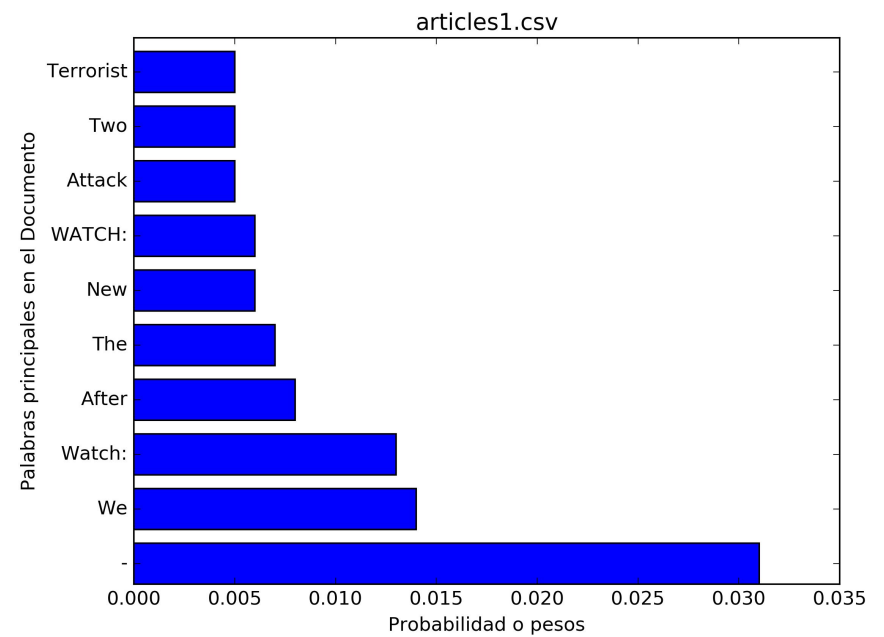
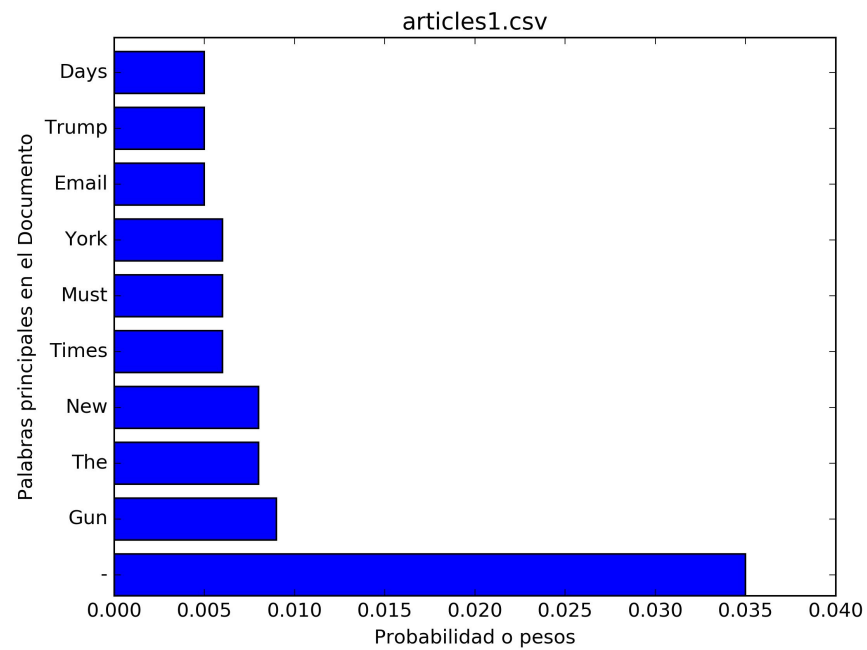
```
c = 49999 / 49999
```

```
0.048*"- " + 0.016*"The" + 0.013*"Donald" + 0.013*"New" + 0.012*"Trump" + 0.010*"York" + 0.010*"Times" + 0.009*"Paul" +  
0.006*"Because" + 0.006*"Anti-Trump"  
0.034*"- " + 0.020*"Trump" + 0.013*"White" + 0.012*"Texas" + 0.011*"House" + 0.010*"Illegal" + 0.010*"Border" + 0.007*"The" +  
0.007*"New" + 0.007*"After"  
0.044*"- " + 0.037*"Hillary" + 0.036*"Clinton" + 0.014*"Trump" + 0.010*"The" + 0.010*"New" + 0.007*"Times" + 0.007*"York" +  
0.007*"Donald" + 0.005*"Clinton's"  
0.033*"- " + 0.016*"Cruz" + 0.014*"Ted" + 0.010*"Donald" + 0.009*"President" + 0.009*"Is" + 0.008*"Bernie" + 0.007*"Trump" +  
0.006*"Control" + 0.006*"Sanders"  
0.032*"- " + 0.020*"Islamic" + 0.018*"Trump" + 0.017*"State" + 0.010*"American" + 0.009*"John" + 0.008*"To" + 0.007*"The" +  
0.006*"New" + 0.006*"Court"  
0.038*"- " + 0.012*"Trump" + 0.010*"The" + 0.009*"New" + 0.007*"Rubio" + 0.007*"York" + 0.007*"Ryan" + 0.006*"Open" +  
0.006*"Times" + 0.006*"Is"  
0.192*"Breitbart" + 0.102*"- " + 0.021*"Trump" + 0.006*"Is" + 0.005*"Not" + 0.005*"Obama" + 0.005*"Will" + 0.005*"Trump:" +  
0.005*"The" + 0.004*"GOP"  
0.036*"To" + 0.026*"- " + 0.011*"Man" + 0.010*"Migrant" + 0.007*"Terror" + 0.007*"The" + 0.006*"New" + 0.006*"Following" +  
0.006*"Mexican" + 0.006*"Police"  
0.031*"- " + 0.014*"We" + 0.013*"Watch:" + 0.008*"After" + 0.007*"The" + 0.006*"New" + 0.006*"WATCH:" + 0.005*"Attack" +  
0.005*"Two" + 0.005*"Terrorist"  
0.035*"- " + 0.009*"Gun" + 0.008*"The" + 0.008*"New" + 0.006*"Times" + 0.006*"Must" + 0.006*"York" + 0.005*"Email" +  
0.005*"Trump" + 0.005*"Days"
```

# Resultados



# Resultados





# Non-negative Matrix Factorization

1. Imagine if you wanted to decompose a term-document matrix,
2. each column represented a document,
3. each element in the document represented the weight of a certain word
4. The weight might be the raw count, or the tf-idf weighted count ,etc.

$$\mathbf{A} = \begin{matrix} & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 & d_7 \\ \begin{matrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \\ t_9 \end{matrix} & \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{matrix}$$

Terms	Documents
T1: Bab(y,ies,y's)	D1: Infant & Toddler First Aid
T2: Child(ren's)	D2: Babies & Children's Room (For Your Home )
T3: Guide	D3: Child Safety at Home
T4: Health	D4: Your Baby's Health & Safety : From Infant to Toddler
T5: Home	D5: Baby Proofing Basics
T6: Infant	D6: Your Guide to Easy Rust Proofing
T7: Guide	D7: Beanie Babies Collector's Guide
T8: Safety	
T9: Toddler	

Figures from Amy Langville Carl Meyer, SIAM-SEAS-Charleston Slides 3/25/2005

# Non-negative Matrix Factorization

1. NMF (Nonnegative Matrix Factorization) is a matrix factorization method where we constrain the matrices to be nonnegative.
2. Therefore, there is no guarantee that we can recover the original matrix, but the best approximation.
3. Suppose we factorize  $X$  into the matrix product  $X \sim WH$  (approximate)

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_k \end{bmatrix} \quad W = \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_k \end{bmatrix} \quad H = \begin{bmatrix} h_1 \\ h_2 \\ \dots \\ h_k \end{bmatrix}$$

# Non-negative Matrix Factorization

1. What happens when we decompose this into two matrices?
2. Imagine if the documents came from news articles.
3. The word “eat” would be likely to appear in food-related articles,
4. And therefore co-occur with words like “tasty” and “food”.
5. Therefore, these words would probably be grouped together into a “food” component vector,
6. And each article (document) would have a certain weight of the “food” topic.

# Non-negative Matrix Factorization

1. We can interpret as a weighted sum of some components (basis), where each row in H is a component and each row in W a weight
2. Weighting documents by words sort of.
3. Essentially, NMF decomposes each data point into an overlay of certain components.
4. Similar to PCA, we can factor into a lower dimensional form, and approximate the basis components.

components

$$x_i = [w_{i1} \ w_{i2} \ \dots \ w_{ik}] \times \begin{bmatrix} h_1 \\ h_2 \\ \dots \\ h_k \end{bmatrix} = \sum_{j=1}^k w_{ij} \times h_j$$

$w_i$ : weights

# Non-negative Matrix Factorization

1.  $\mathbf{A}$  (Document-word matrix)—input that contains which words appear in which documents.
2.  $\mathbf{W}$  (Basis vectors)—the topics (clusters) discovered from the documents.
3.  $\mathbf{H}$  (Coefficient matrix)—the membership weights for the topics in each document.
4. We calculate  $\mathbf{W}$  and  $\mathbf{H}$  by optimizing over an objective function updating both  $\mathbf{W}$  and  $\mathbf{H}$  **iteratively** until convergence.


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

$$W_{ic} \leftarrow W_{ic} \frac{(\mathbf{AH})_{ic}}{(\mathbf{WHH})_{ic}} \quad H_{cj} \leftarrow H_{cj} \frac{(\mathbf{WA})_{cj}}{(\mathbf{WWH})_{cj}}$$

# Non-negative Matrix Factorization

1. Then an NMF decomposition of the term-document matrix would yield components that could be considered “topics”,
2. And decompose each document into a **weighted sum of topics**.
3. Why the underlying components (topics) and their weights should be non-negative? We cannot interpret what it means to have a “negative” weight of the food topic.
4. NMF is that it naturally produces sparse representations.
5. This makes sense in the case of topic modeling: documents generally do not contain a large number of topics.
6. Reference: Keita Kurita, A practical Introduction to NMF.


# 1 million head titles dataset: Australian ABC News



Dataset

## A Million News Headlines

News headlines published over a period of 15 years.

 Rohk · updated a year ago

183 voters

share

Data

Overview

Kernels (32)

Discussion (5)

Activity

Download (19 MB)

New Kernel

Tags

linguistics

news agencies

sociology

historiography

medium

featured

Description

### Context

This contains data of news headlines published over a period of 15 years.

Sourced from the reputable Australian news source ABC (Australian Broadcasting Corp.)

Agency Site: <http://www.abc.net.au/>

### Content

Format: CSV ; Single File

1. publish\_date: Date of publishing for the article in yyyyMMdd format
2. headline\_text: Text of the headline in Ascii , English , lowercase

Start Date: 2003-02-19 End Date: 2017-12-31

Total Records: **1,103,665**

Feed Code: w3-event-abcaus; Si.gh.rank: BJL

DOI:10.1145/2133806.2133826

**Surveying a suite of algorithms that offer a solution to managing large document archives.**

BY DAVID M. BLEI

## Probabilistic Topic Models

AS OUR COLLECTIVE knowledge continues to be digitized and stored—in the form of news, blogs, Web pages, scientific articles, books, images, sound, video, and social networks—it becomes more difficult to find and discover what we are looking for. We need new computational tools to help organize, search, and understand these vast amounts of information.

Right now, we work with online information using two main tools—search and links. We type keywords into a search engine and find a set of documents related to them. We look at the documents in that set, possibly navigating to other linked documents. This is a powerful way of interacting with our online archive, but something is missing.

Imagine searching and exploring documents based on the themes that run through them. We might “zoom in” and “zoom out” to find specific or broader themes; we might look at how those themes changed through time or how they are connected to each other. Rather than finding documents through keyword search alone, we might first find the theme that we are interested in, and then examine the documents related to that theme.

For example, consider using themes to explore the complete history of the New York Times. At a broad level, some of the themes might correspond to the sections of the newspaper—foreign policy, national affairs, sports. We could zoom in on a theme of interest, such as foreign policy, to reveal various aspects of it—Chinese foreign policy, the conflict in the Middle East, the U.S.’s relationship with Russia. We could then navigate through time to reveal how these specific themes have changed, tracking, for example, the changes in the conflict in the Middle East over the last 50 years. And, in all of this exploration, we would be pointed to the original articles relevant to the themes. The thematic structure would be a new kind of window through which to explore and digest the collection.

But we do not interact with electronic archives in this way. While more and more texts are available online, we simply do not have the human power to read and study them to provide the kind of browsing experience described above. To this end, machine learning researchers have developed *probabilistic topic modeling*, a suite of algorithms that aim to discover and annotate large archives of documents with thematic information. Topic modeling algorithms are statistical methods that analyze the words of the original texts to discover the themes that run through them, how those themes are connected to each other, and how they change over

### » key insights

- Topic models are algorithms for discovering the main themes that pervade a large and otherwise unstructured collection of documents. Topic models can organize the collection according to the discovered themes.
- Topic modeling algorithms can be applied to massive collections of documents. Recent advances in this field allow us to analyze streaming collections, like you might find from a Web API.
- Topic modeling algorithms can be adapted to many kinds of data. Among other applications, they have been used to find patterns in genetic data, images, and social networks.