Topic Modelling
(Non-negative Matrix Factorization)

SCHOOL OF INFOCOMM

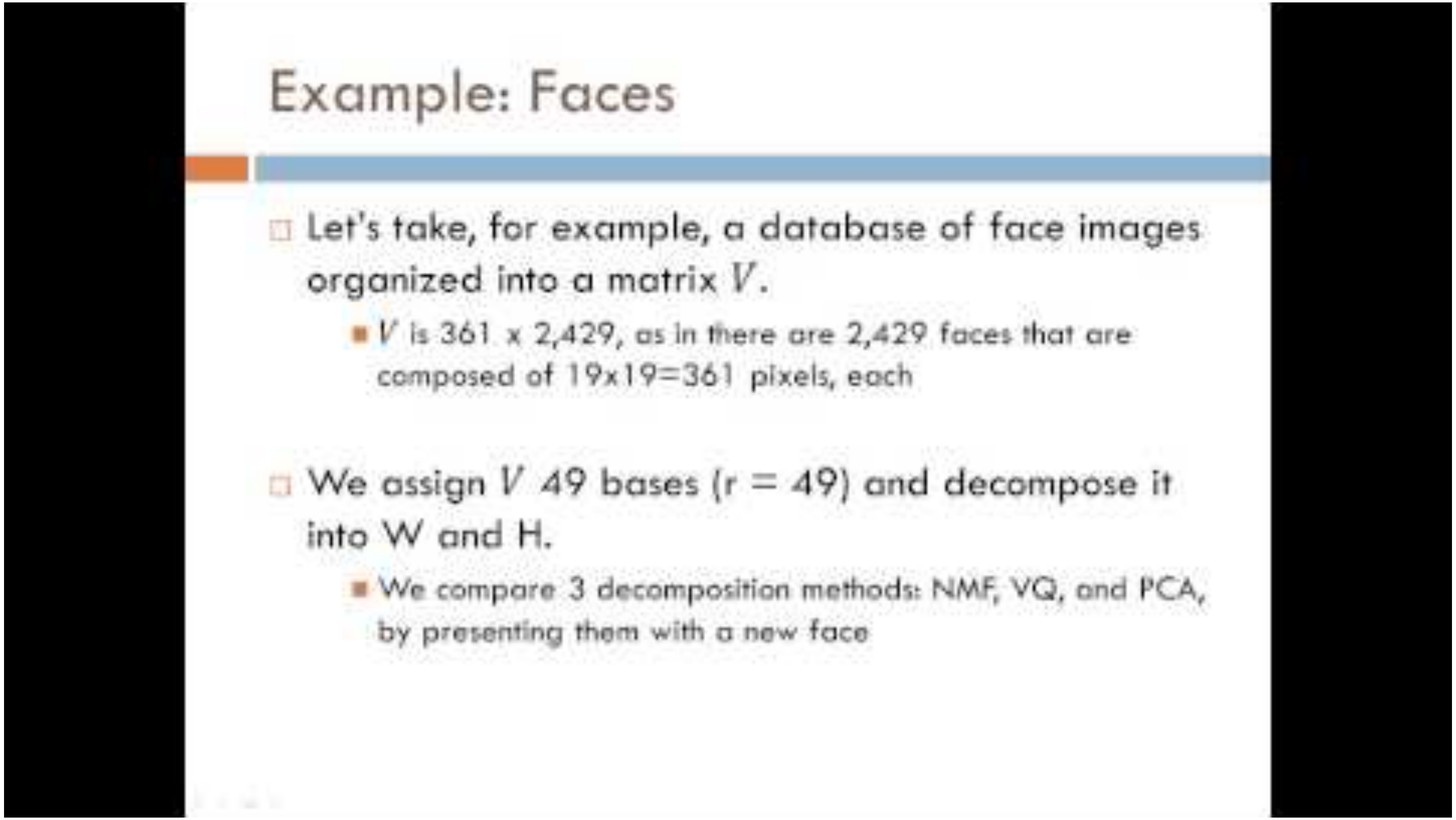
Recap

- Topic modelling can be considered as dimension reduction processes
  - Go from a “word space”, which may be 1000s of dimensions, to a smaller dimensionality “combinations of words space.” We called this a “topic space”

- Latent Dirichlet Allocation, a probabilistic and generative model , is one method to determine the "topic space" of a large body of text. It assumes that documents are made up of many topics in parts

- There are mathematically rigorous ways to accomplish the same task
  - Principal component analysis
  - Matrix factorization

Learning Video



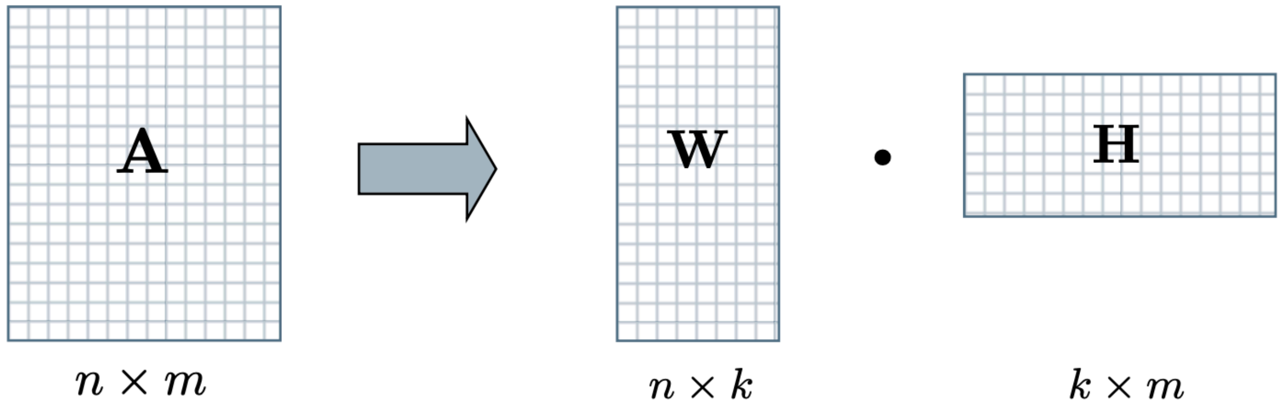
https://youtu.be/UQGEB3Q5-fQ

Non-negative Matrix Factorization (NMF)

Nonnegative matrix factorization (NMF) is an unsupervised family of algorithms that performs dimension reduction and factors analysis

Widely used tool for the analysis of high dimensional data as it automatically extracts sparse and meaningful features from a set of nonnegative data vectors

Learn and produce a "parts-based" representation of the larger data



Data Matrix
(Rows = Features, Cols = Objects)

Basis Vectors
(Rows = Features)

Coefficient Matrix
(Cols = Objects)

image: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.702.4867&rep=rep1&type=pdf

NMF Applications - Astronomy

Data : Spectroscopic observation and direct image observation

Factors: common properties of astronomical objects and post-process the astronomical observations.

Results: Reveal the faint exoplanets

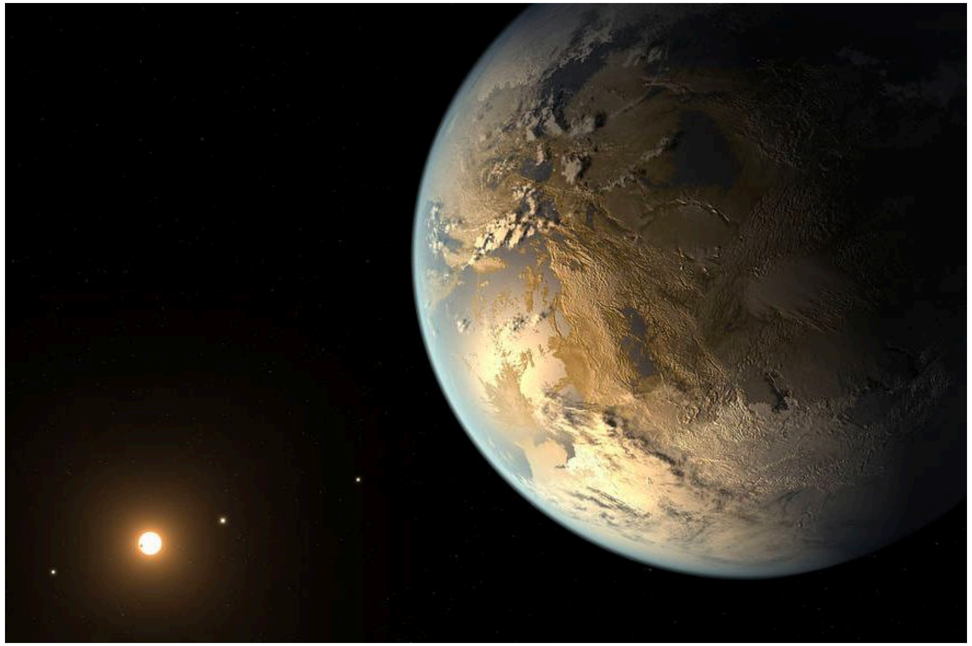


image: NASA

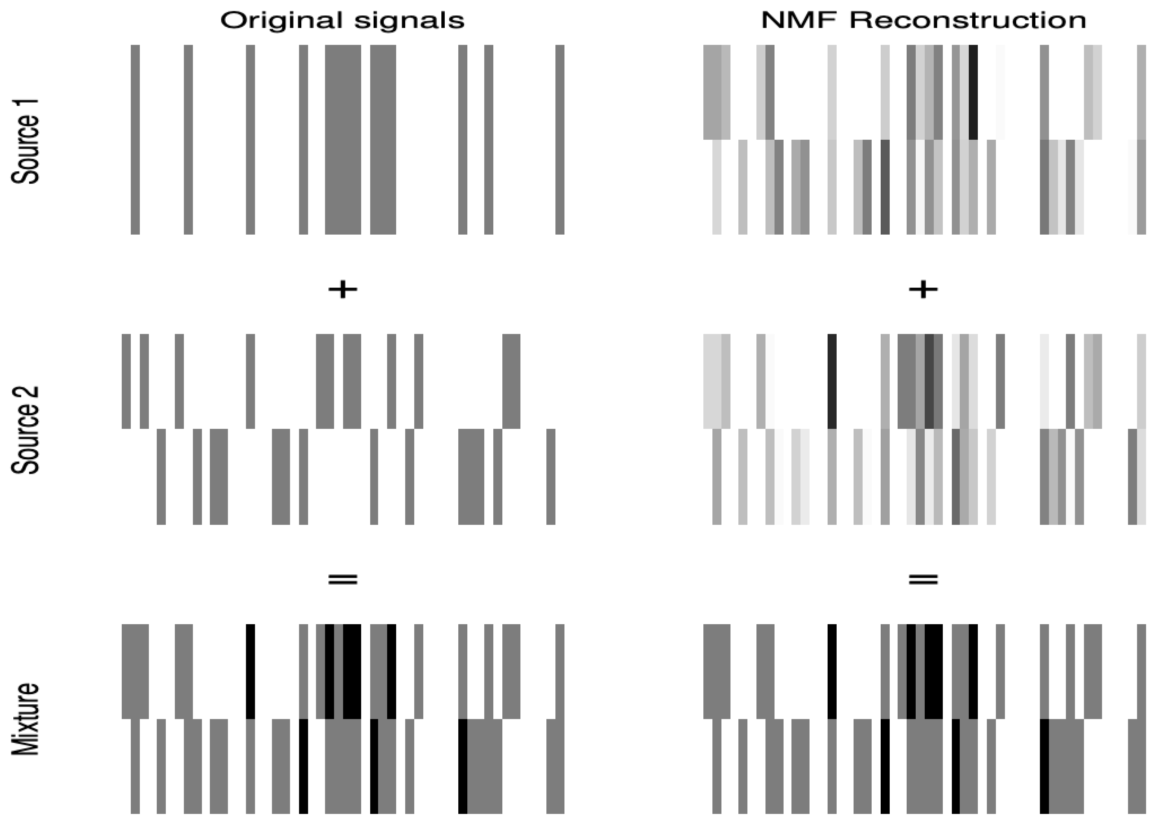
More info: Non-negative Matrix Factorization: Robust Extraction of Extended Structures
https://arxiv.org/pdf/1712.10317.pdf

NMF Applications – Sound Processing

Input: Noisy sound multiple sources

Factors: sources of sounds – speech, noise sources. (jackhammer noise, bus/street noise, combat noise, and speech babble noise)

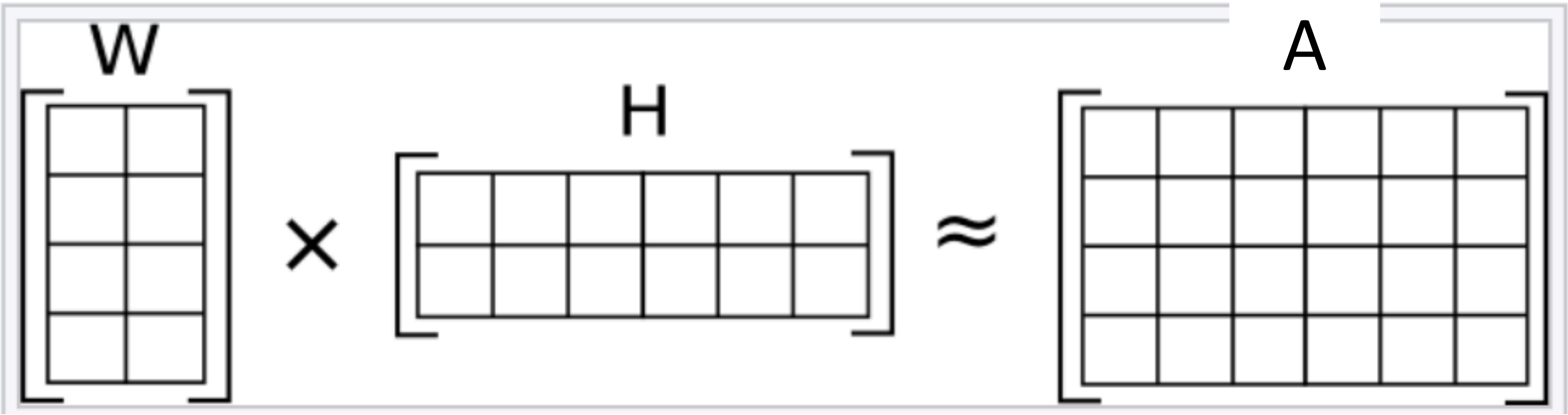
Results: Audio sources separation



More info:
SPEECH DENOISING USING NONNEGATIVE MATRIX FACTORIZATION WITH PI
https://paris.cs.illinois.edu/pubs/wilson-icassp08.pdf

Adaptive Noise Reduction for Sound Event Detection Using Subband-Weighted NMF
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6679307/

Given a non-negative matrix A, find non-negative matrix factors Wand H such that V~WH



(Basis vectors) — k topics discovered from n documents

H (Coefficient matrix) — the membership weights for the topics in each document

A - (Document-word matrix).
***** in some documentation, the notation V is also used***

Image: Wikpedia

Topic modelling with NMF

	research	school	education	disease	patient	health	budget	finance	banking	bonds
document 1										
document 2										
document 3										
document 4										
document 5										
document 6										

A :
6 documents with 10 terms.
The scoring can be count or TFIDF

Factor **W**
Weights for 6 documents relative to 3 topics

	topic 1	topic 2	topic 3
document 1			
document 2			
document 3			
document 4			
document 5			
document 6			

Factor **H**
Weights for 10 terms relative to 3 topics

	research	school	education	disease	patient	health	budget	finance	banking	bonds
topic 1										
topic 2										
topic 3										

Scikit Learn NFM


```
nmf_model = NMF(n_components=5, init='random',random_state=0 )

A = TfidfVectorizer().fit_transform(documents)
H = nmf_model.components_
W = nmf_model.fit_transform(A)
```

n_components : number of topics
init : initialise the matrics with sqrt(X.mean() / n_components)
random_state : random seed generator

What is in the H Factor

H factor (**nmf_model.components_**) contains the term weight relative to each topic. Each row is a topic. Each column is a unique term

	research	school	education	disease	patient	health	budget	finance	banking	bonds
topic 1										
topic 2										
topic 3										

```
[[0.4642 0.3434 0.     0.4074 0.     0.4814 0.     0.4702 0.     0.    ]
 [0.     0.2899 0.3642 0.     0.     0.     0.3837 0.     0.     0.    ]
 [0.     0.6138 0.     0.     1.0393 0.     0.     0.     1.0393 1.0393]
 [0.388  0.0317 0.     0.55   0.     0.3389 0.     0.3708 0.     0.    ]
 [0.     0.     1.7546 0.     0.     0.     1.4841 0.     0.     0.    ]]
```

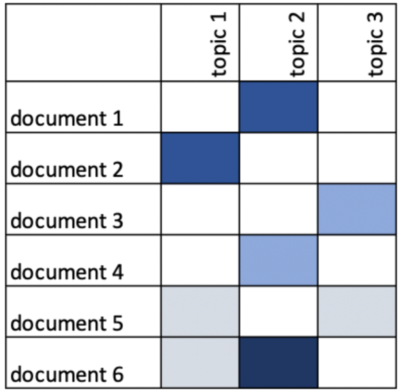
The example above has 5 topics with 10 terms

Sorting the values in each row gives a ranking of the terms for each topic

What is in the W Factor

W contains the document membership weights across the k topics. Each row is a different document ; Each column is a topic

```
[[0.0000e+00 1.3293e+00 0.0000e+00 0.0000e+00 9.5947e-02]
 [7.9892e-01 0.0000e+00 1.4538e-08 2.8014e-01 0.0000e+00]
 [0.0000e+00 1.4770e-04 5.2578e-01 5.7715e-06 0.0000e+00]]
```



The example above has 3 documents with 5 topics

Sorting the values in across each row tells us which is the dominant topic(s) in a document.

Sorting the values in each column (topic) will provide a ranking of the documents relevant to a topic

Exercise 1 - Getting familiar with NMF

Refer to Jupyter Notebook: ex1_nmf_basic.ipynb

Vectorize a 'toy' corpus

Apply NMF

Examine the W and H

Exercise 2 - Load corpus and vectorised to get martix A

Refer to Jupyter Notebook:

ex2-nmf_text_preprocessing.ipynb

Read in a text document

Remove stop words
Stemming

Prepare term-document metric (A) using count using Count or TFIDF

Save the document, terms, term-document matrix(A)for future use

Exercise 3 - Apply NMF to a Document-Term matrix and examine relationship between A, H and W

Refer to Jupyter Notebook:

ex3-nfm_apply_topic_models.ipynb

Reload the document, terms, term-document matrix (A)

Apply NMF to get W and H

Examine the shape

Save the document, terms, document-term matrix, NMF model for future use

Exercise 4 - Examine topic-terms distribution

Refer to Jupyter Notebook: ex4-nfm_topic_terms_distribution.ipynb

Reload A, H, W from previous step

Apply NMF

Reverse sort H row-rise
(topic-terms in H)

Plot distribution of terms using horizonal bar charts

Examine topic-term distribution

	research	school	education	disease	patient	health	budget	finance	banking	bonds
topic 1										
topic 2										
topic 3										

```python
import numpy as np
def get_descriptor( terms, H, topic_index, top ):
    # reverse sort the values to sort the indices
    top_indices = np.argsort( H[topic_index,:] )[::-1]
    # now get the terms corresponding to the top-ranked indices
    top_terms = [ ]
    for term_index in top_indices[0:top]:
        top_terms.append( terms[term_index] )
    return top_terms
```

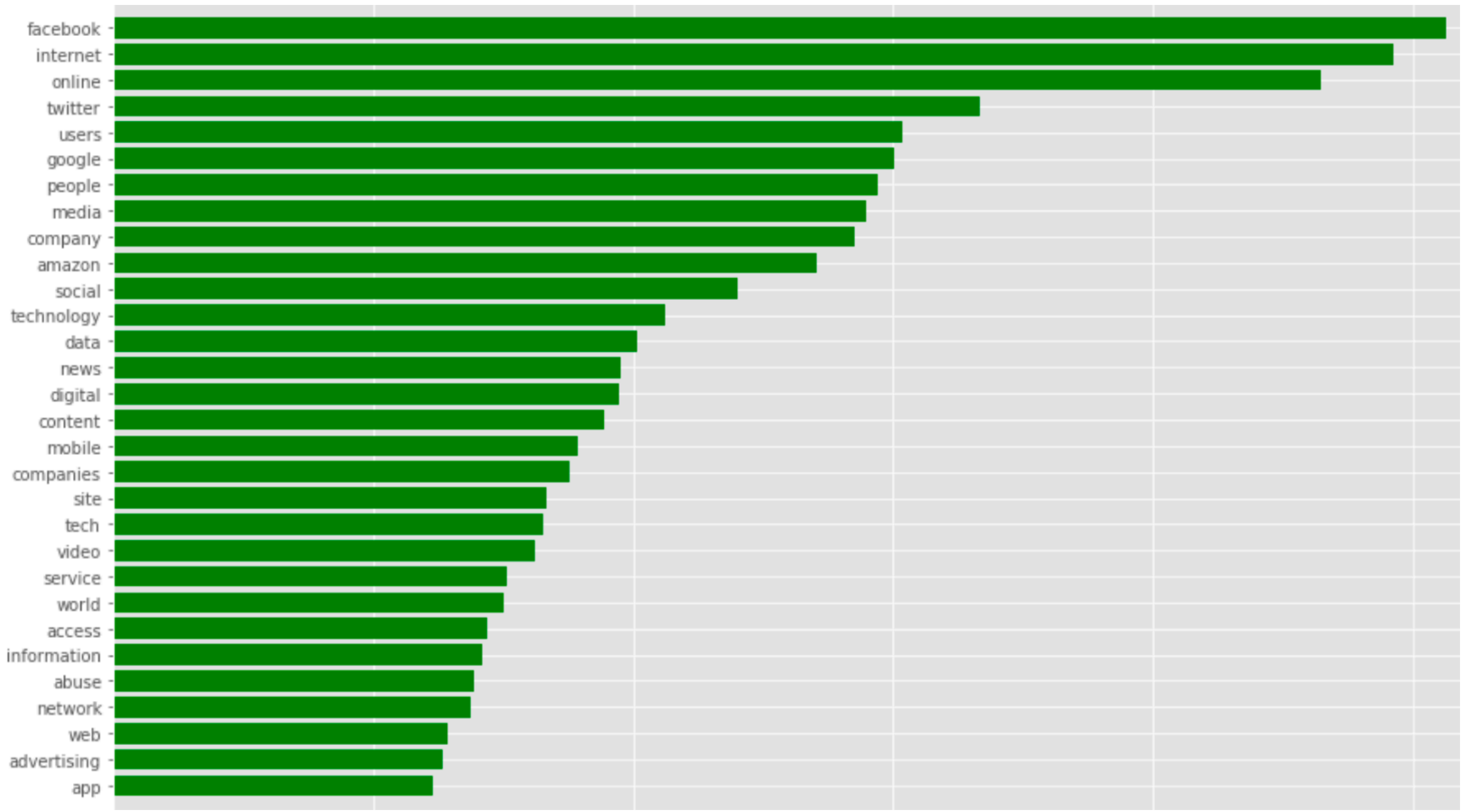
```python
descriptors = [ ]
for topic_index in range(k):
    descriptors.append( get_descriptor( terms, H, topic_index, 6 ) )
    str_descriptor = ", ".join( descriptors[topic_index] )
    print("Topic %02d: %s" % ( topic_index+1, str_descriptor ) )
```

```
Topic 01: eu, uk, brexit, britain, european, leave
Topic 02: trump, clinton, donald, republican, campaign, president
Topic 03: film, films, movie, star, director, hollywood
Topic 04: league, season, leicester, goal, premier, united
```

Examine topic-term distribution

```
Topic 07: album, music, band, song, pop, songs, rock, love, sound, bowie
Topic 08: facebook, internet, online, twitter, users, google, people, media, company, amazon
Topic 09: labour, party, corbyn, cameron, referendum, vote, voters, campaign, johnson, minister
Topic 10: women, abortion, woman, men, cancer, female, ireland, girls, rights, northern
```

Distribution of terms in topic 8



Exercise 5
Finding documents related to a topic

Refer to Jupyter Notebook:

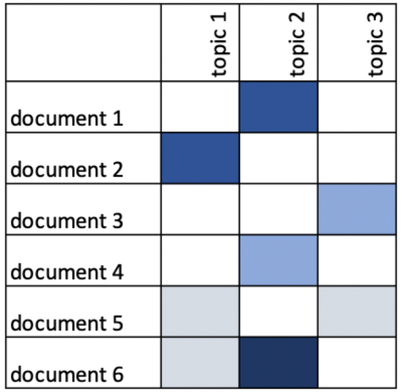
ex5-nfm_apply_topic_models.ipynb

Reload the document, terms, term-document matrix (A)

Apply NMF to get W and H

Extract top documents for a given topic (W)

To Get The Most Relevant Document for a Topic




```python
def get_top_snippets( all_snippets, W, topic_index, top ):
    # reverse sort the values to sort the indices
    top_indices = np.argsort( W[:,topic_index] )[::-1]
    # now get the snippets corresponding to the top-ranked indices
    top_snippets = []
    for doc_index in top_indices[0:top]:
        top_snippets.append( all_snippets[doc_index] )
    return top_snippets
```

```python
topic_snippets = get_top_snippets( snippets, W, 1, 10 )
for i, snippet in enumerate(topic_snippets):
    print("%02d. %s" % ( (i+1), snippet ) )
```

```
01. Donald Trump: money raised by Hillary Clinton is 'blood money' –
02. Second US presidential debate – as it happened Here's how searche
03. Trump campaign reportedly vetting Christie, Gingrich as potential
04. Donald Trump hits delegate count needed for Republican nomination
05. Trump: 'Had I been president, Capt Khan would be alive today' – a
```

What is the right number of topics?

- Topic modelling provides us with methods to organize and summarize large collections of textual information, but gives **no guarantee** on the interpretability of their output

- Selection of an appropriate number of topics (k : n_components) is key to successful application of topic modelling
  - *k that is too low will generate topics that are overly broad*
  - *k that is too high will result in too much topics that may overlay*

Making sense of the terms

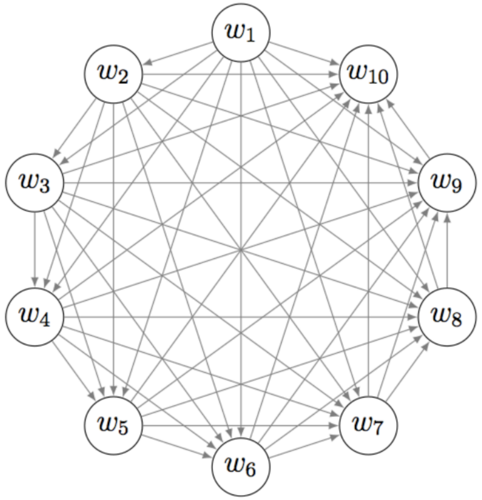
Topic 07: album, music, band, song, pop, songs, rock, love, sound, bowie
Topic 08: facebook, internet, online, twitter, users, google, people, media, company, amazon
Topic 09: labour, party, corbyn, cameron, referendum, vote, voters, campaign, johnson, minister
Topic 10: women, abortion, woman, men, cancer, female, ireland, girls, rights, northern

Within a topic, how can we tell that the words within are "similar /relate / close"

- To what degree is "facebook" is similar to "media"
- To what degree is "internet" is similar to "twitter"
- Overall, are all the pairs of words "similiar"

Topic Coherence

Topic



- For each pair of words
  Find the **pairwise score**
- **Topic Coherence** = average of all the pairwise score

Topic Coherence Measure
- UCI
- UIMass
- Word2Vec Similarities

details

Topic Coherence Measure – UCI

Extrinsic **UCI** – measures the **probability** of a **pair of words** co-occurring in the same document against an external corpus

$$C_{UCI} = \frac{1}{{}^{N}C_2} \sum_{j=2}^{N} \sum_{i=1}^{j-1} log\left(\frac{P(w_j,\ w_i)\ +\ \epsilon}{P(w_i)\ P(w_j)}\right)$$

<u>Notes</u>

N = # of Top words from a Topic. eg; N=10

*p(wi,wj) is the probability of seeing both wi and wj co-occurring in a **random document.***

*p(w) represents the probability of seeing wi in a **random document***

*Those probabilities are empirically estimated from **an external dataset** such as Wikipedia*

Topic Coherence Measure – UMass

Intrinsic **UMass** – measures **the number of documents** which a pair of words appears relative to the number of documents one word appears

$$C_{UMass} = \frac{1}{{}^{N}C_2} \sum_{j=2}^{N} \sum_{i=1}^{j-1} log\left(\frac{P(w_j, w_i) + \epsilon}{P(w_i)}\right)$$

<u>Notes:</u>

N = number of top words from a topic. eg; N=10

P(wi) = number of documents that the word w occurring divide by total number of documents in the ***corpus***

P(wj, wi) = number of documents that pair of words (wj and wi) occurring divide by total number of documents in the ***corpus***

Topic Coherence Measure – TC-W2V

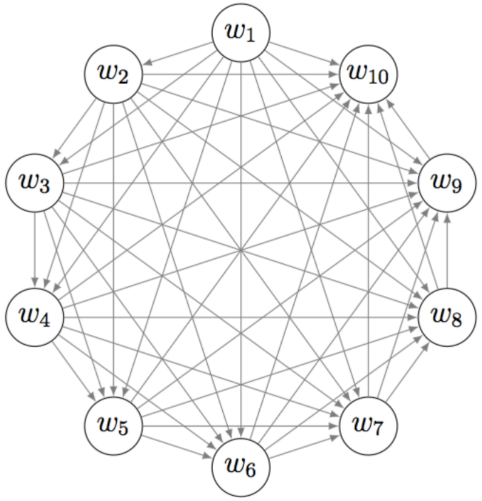
Topic Coherence Word2Vec (**TC-W2V)** – measures how **semantically close** are the words by measuring the similarity (e.g. cosine difference) between **pairs of term vectors** using Word2Vec model

$$\text{TC-W2V} = \frac{1}{\binom{N}{2}} \sum_{j=2}^{N} \sum_{i=1}^{j-1} similarity(wv_j, wv_i)$$

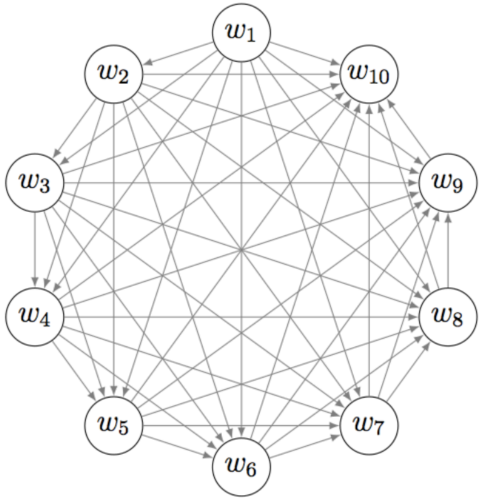
Notes:
similarity() is a method available in Word2Vec model implementation

Using Overall Coherence to Determine the Optimal Number of Topics

Topic #1



Topic #2



- For each pair of words
  Find the pairwise score
- Topic #1 Coherence = average of all the pairwise score

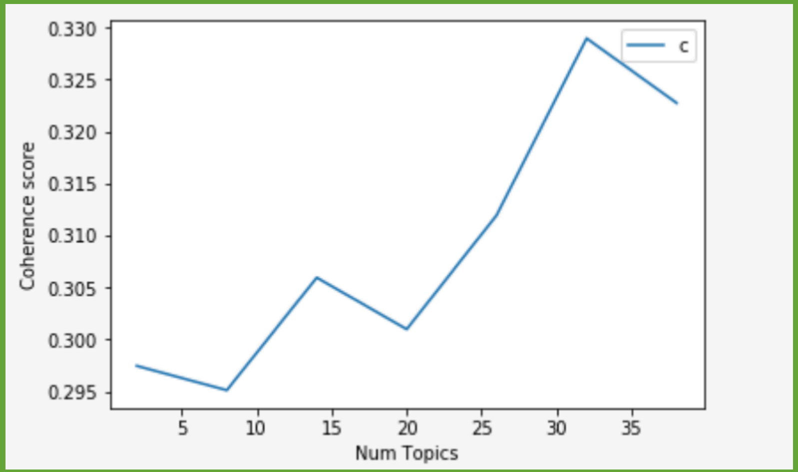
- For each pair of words
  Find the pairwise score
- Topic #2 coherence = average of all the pairwise score

Overall coherence = Average of all topics coherence

e.g. (Topic 1 Coherence + Topic 2 Coherence) / 2

Highest overall coherence → best number of topics

Exercise 6 - Find the best number of topics using TC-W2V



This exercise can be defer to next lesson in order to understand Word2Vec

Refer to Jupyter Notebook:
ex6-nmf_determine_optimal_topics.ipynb

Reload the document-term matrix, terms, and original document

Generate word2vec representation

Create topic models for a range of k

For each topic in each k, calculate topic coherence

For each k, calculate overall coherence

Choose the k that is associated with the highest mean coherence

t

Rule of Thumbs

- NMF tends to outperform LDA on small documents or documents associated with non-mainstream domains
- NMF performs differently with TF-IDF vs Count Vectorizer. The former yields better topics
- The original text structure and pre-processing steps affect the outcome of topic modelling
- Use topic modelling as an exploratory tool to assist with human interpretation

References

- The Why and How of Nonnegative Matrix Factorization, https://arxiv.org/pdf/1401.5226.pdf
- Algorithms for Non-negative Matrix Factorization, https://papers.nips.cc/paper/1861-algorithms-for-non-negative-matrix-factorization.pdf
- Quick introduction to nonnegative matrix factorization, http://heather.cs.ucdavis.edu/NMFTutorial.pdf
- Learning the parts of objects by non-negative matrix factorization. http://www.columbia.edu/~jwp2128/Teaching/E4903/papers/nmf_nature.pdf
- How many topics? Stability analysis for topic models, http://derekgreene.com/papers/greene14topics.pdf
- Using Word Embedding to Evaluate the Coherence of Topics from Twitter Data, http://terrierteam.dcs.gla.ac.uk/publications/fang_sigir_2016_we.pdf
- How to measure topic coherence, https://labs.imaginea.com/how-to-measure-topic-coherence
- Evaluate topic coherence, https://arxiv.org/pdf/1403.6397.pdf
- An analysis of the coherence of descriptors in topic modeling, https://www.insight-centre.org/sites/default/files/publications/15.010_eswa2014_final_submit.pdf
- Nonnegative matrix factorization for interactive topic modeling and document clustering, https://www.cc.gatech.edu/~hpark/papers/nmf_book_chapter.pdf