NLP basics: From text to vectors

Setup
Github: https://github.com/bpben/pydata_nlp_workshop

Notebook: https://github.com/bpben/pydata_nlp_workshop/blob/master/nlp_workshop.ipynb

Open using Google Collaboratory (preferred):
https://colab.research.google.com/github/bpben/pydata_boston/blob/master/notebooks/nlp_workshop.ipynb

You need to upload the data to your collab instance:
https://github.com/bpben/pydata_nlp_workshop/blob/master/movie_reviews_subset.pkl

About me

Associate Director, Data Science

PhD, Policy Analysis

Website: https://benbatorsky.com/

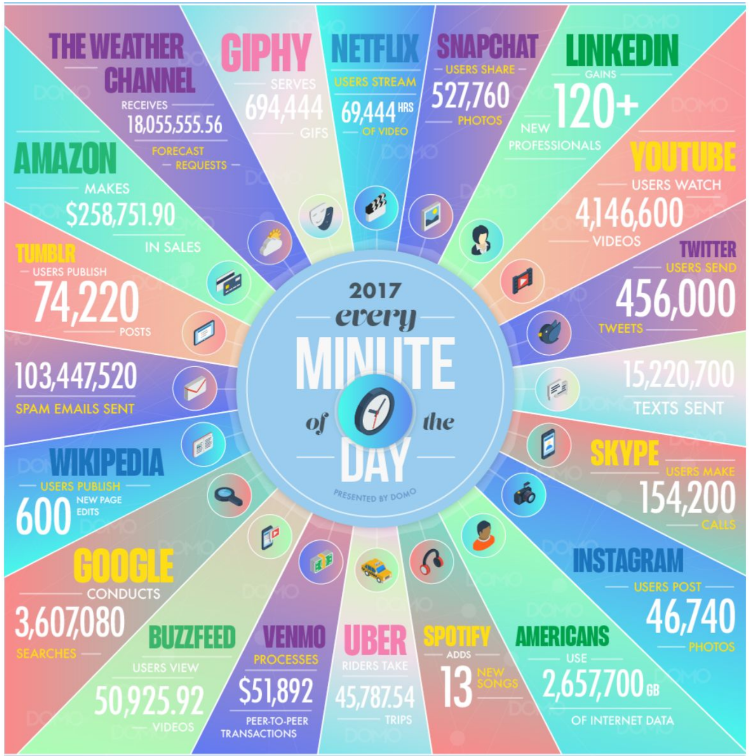
Github: https://github.com/bpben

MIT MANAGEMENT SLOAN SCHOOL

Food Supply Chain Analytics and Sensing Group

Global pilots of risk-based food safety testing technology

Explosion of data...unstructured data, that is



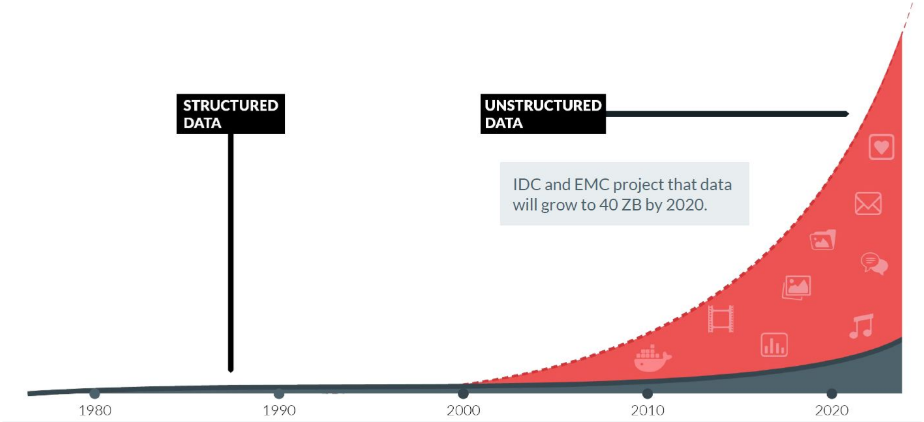
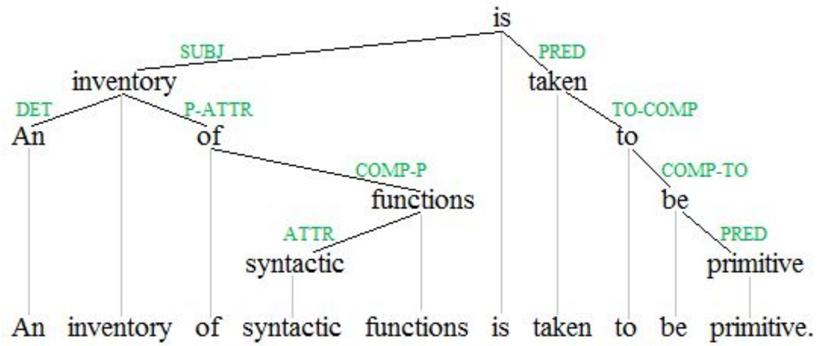


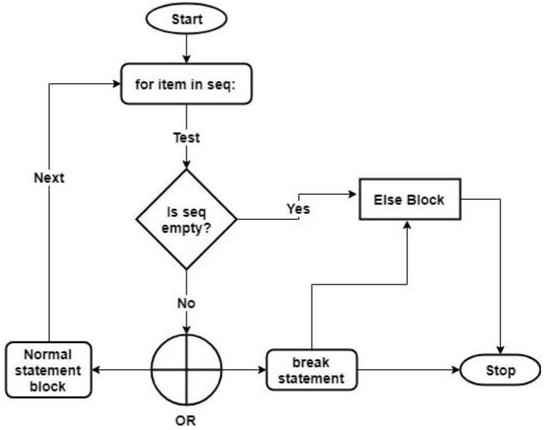
Chart:https://www.datanami.com/2017/02/01/solving-storage-just-beginning-minio-ceo-periasamy/
Data: IDC Structured Versus Unstructured Data: The Balance of Power Continues to Shift, March 2014

What is Natural Language?

"A language that has developed naturally in use (as contrasted with an artificial language or computer code)." (Oxford Dictionary definition)



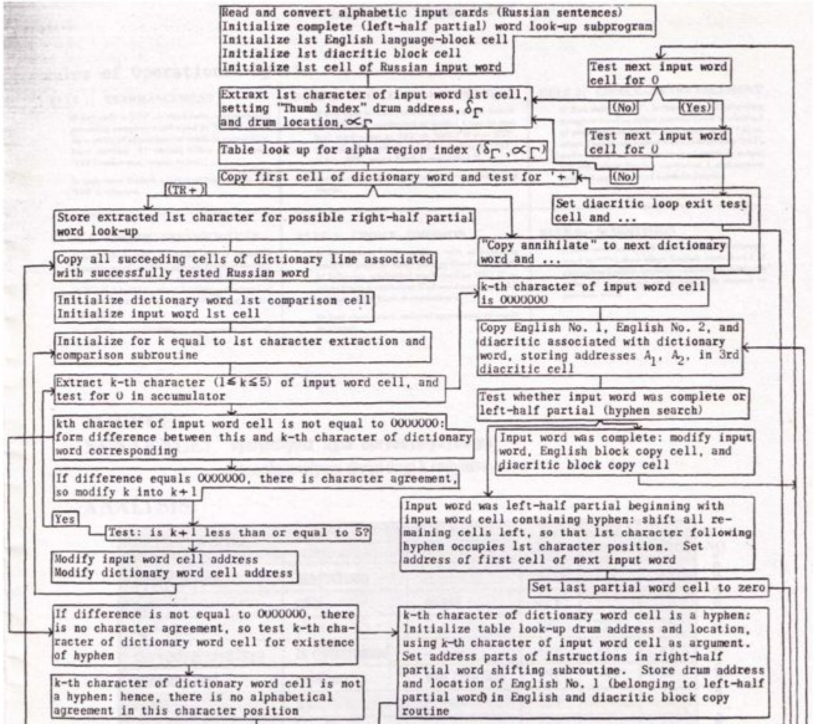
https://en.wikipedia.org/wiki/Dependency_grammar



https://www.techbeamers.com/python-for-loop/

How do we process language into text?

1950



2013

https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Now we can do things like this

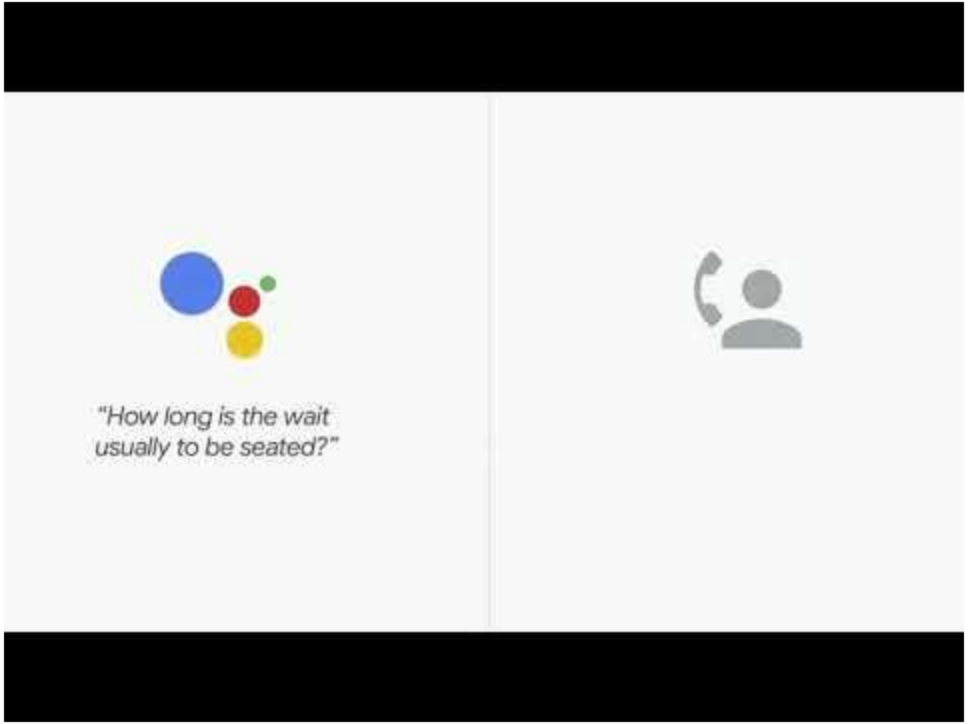
Write with Transformer from HuggingFace:
https://transformer.huggingface.co/doc/distil-gpt2

I am attending a PyData meetup on **the weekend at 6pm (2pm Pacific time, 6pm GMT). As part of the PyData session, there will be a presentation at the event to give attendees ideas for how to get started and build your own Python applications.**

Written by Transformer · transformer.huggingface.co

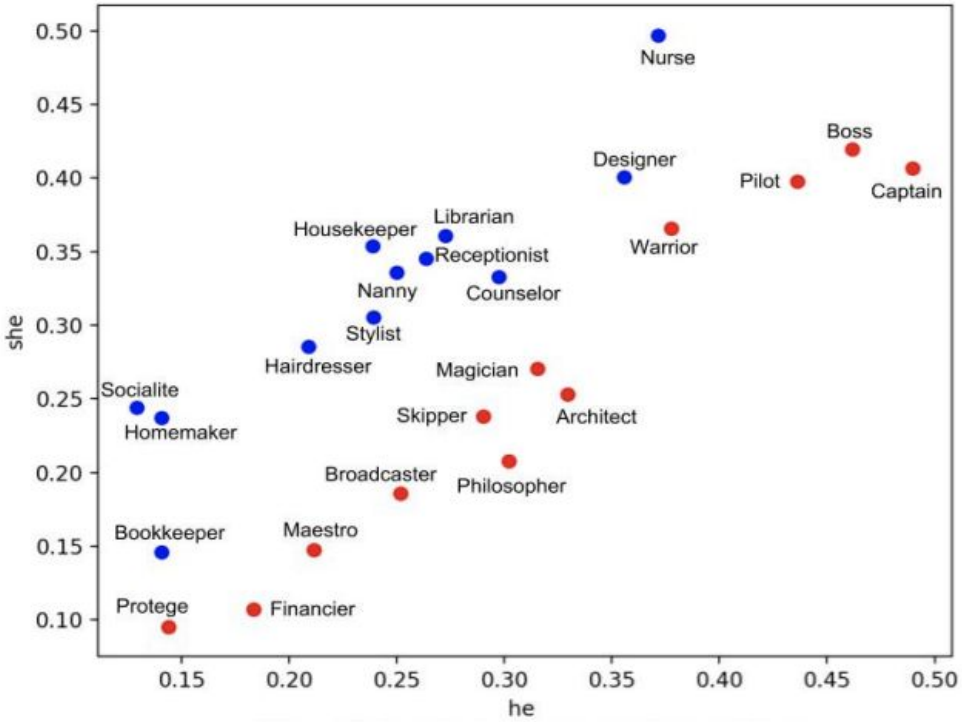
And this:

Google Assistant making a reservation



Though also, this:

Word vector similarity between gender words and occupations



[Exploring and Mitigating Gender Bias in GloVe Word Embeddings]

NLP libraries

Comparison of Python NLP libraries Pros and Cons

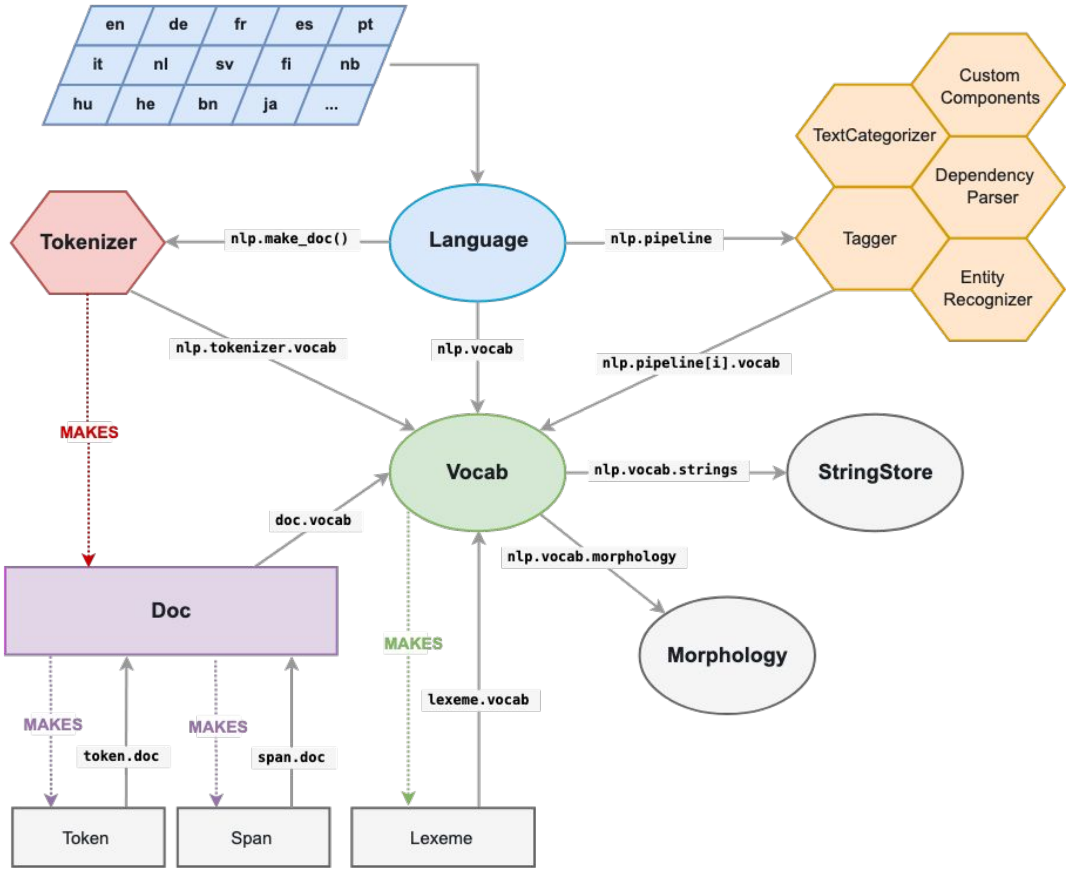
	⊕ PROS	⊖ CONS
Natural Language ToolKit	+ The most well-known and full NLP library + Many third-party extensions + Plenty of approaches to each NLP task + Fast sentence tokenization + Supports the largest number of languages compared to other libraries	- Complicated to learn and use - Quite slow - In sentence tokenization, NLTK only splits text by sentences, without analyzing the semantic structure - Processes strings which is not very typical for object-oriented language Python - Doesn't provide neural network models - No integrated word vectors
spaCy	+ The fastest NLP framework + Easy to learn and use because it has one single highly optimized tool for each task + Processes objects; more object-oriented, comparing to other libs + Uses neural networks for training some models + Provides built-in word vectors + Active support and development	- Lacks flexibility, comparing to NLTK - Sentence tokenization is slower than in NLTK - Doesn't support many languages. There are models only for 7 languages and "multi-language" models
scikit learn NLP toolkit	+ Has functions which help to use the bag-of-words method of creating features for the text classification problems + Provides a wide variety of algorithms to build machine learning models + Has good documentation and intuitive classes' methods	- For more sophisticated preprocessing things (for example, pos-tagging), you should use some other NLP library and only after it you can use models from scikit-learn - Doesn't use neural networks for text preprocessing
gensim	+ Works with large datasets and processes data streams + Provides tf-idf vectorization, word2vec, document2vec, latent semantic analysis, latent Dirichlet allocation + Supports deep learning	- Designed primarily for unsupervised text modeling - Doesn't have enough tools to provide full NLP pipeline, so should be used with some other library (Spacy or NLTK)
Pattern	+ Allows part-of-speech tagging, n-gram search, sentiment analysis, WordNet, vector space model, clustering and SVM + There are web crawler, DOM parser, some APIs (like Twitter, Facebook etc.)	- Is a web miner; can be not enough optimized for some specific NLP tasks
Polyglot	+ Supports a large number of languages (16-196 languages for different tasks)	- Not as popular as, for example, NLTK or Spacy; can be slow issues solutions or weak community support

Created by ActiveWizards

[Comparison of Top 6 Python NLP Libraries - ActiveWizards — your AI partner]

Tokenization

- Token: A useful semantic unit
- SpaCy's Language modules
  - Language-specific pipeline
    - Base model: Tokenizer
    - Trained model: Entity recognizer, part of speech tagger, etc
- Document -> Span -> Token
  - Token->entities, etc
- What is a "useful semantic unit"?



https://spacy.io/

Stemming vs Lemmatization

Goal: Reduce related words to a common "base form"

Stemming

Set of heuristics to transform suffixes. Most of the word stays intact.

"He does natural language processing"

↓

"He doe natural language process"

Implemented in: NLTK, PyStemmer

Lemmatization

Transforms word to their "lemma", or dictionary form. This may change the word entirely.

"He is doing natural language processing"

↓

"He be do natural language process"

Implemented in: NLTK, spaCy

Stop words

- Extremely common words that have little information
    - “the”, “an”, “from”, “to”
- No single list of these
    - Consequences to choosing overly strict/broad

In May 2020, I went to a PyData meetup in Cambridge.

Stop words

- Extremely common words that have little information
  - “the”, “at”, “from”, “to”, “in”
- No single list of these
  - Consequences to choosing overly strict/broad

In May 2020, I went to a PyData meetup in Cambridge.

<Remove stop words (SpaCy)>

May 2020, went PyData meetup Cambridge

Named-Entities

- Named-entity: A real-world named object (e.g. person, place, organization)
  - New York City is different than just an assembly of three words “new”, “york” and “city”
- To identify these
  - Dictionaries
  - Pattern-matching
  - Models

In May 2020, I went to a PyData meetup in Cambridge.

Named-Entities

- Named-entity: A real-world named object (e.g. person, place, organization)
    - New York City is different than just an assembly of three words “new”, “york” and “city”
- To identify these
    - Dictionaries
    - Pattern-matching
    - Models

In May 2020 [DATE], I went to a PyData [ORG] meetup in Cambridge [LOC].

Document-Term and Term-Term matrices

- Information Retrieval: Extract signal from noise
  - Useful representations of the documents/terms
- Document representation in vocabulary space
  - Document - Term Matrix (DTM)
  - DTM = Documents x vocabulary
- Term representation in vocabulary space
  - How often two terms occur in the same document
  - DTM * inverse DTM = Term-Term matrix (TTM)
- We can do some neat things with just these vectors

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Figure 6.2 The term-document matrix for four words in four Shakespeare plays. Each cell contains the number of times the (row) word occurs in the (column) document.

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	
strawberry	0	...	0	0	1	60	19	
digital	0	...	1670	1683	85	5	4	
information	0	...	3325	3982	378	5	13	

Figure 6.5 Co-occurrence vectors for four words in the Wikipedia corpus, showing six of the dimensions (hand-picked for pedagogical purposes). The vector for *digital* is outlined in red. Note that a real vector would have vastly more dimensions and thus be much sparser.

Issues with raw word counts

- NLP: Turn text into information
- Raw word count = each word counted the same
  - “This book is about biology” vs “This book is about history”
- Ways to reduce the noise
  - Reducing to common forms
  - Stripping uninformative words (“the”, “and”)
- More standard way up upweighting important words, discounting unimportant ones

Top ten words for negative movie reviews

```
[('the', 15365),
 ('a', 7548),
 ('and', 6978),
 ('to', 6780),
 ('of', 6402),
 ('is', 4952),
 ('it', 4354),
 ('i', 4248),
 ('in', 4203),
 ('this', 3837)]
```

Term Frequency - Inverse Document Frequency (TF-IDF)

- Term frequency: Count of term (or token) within a document
- Document frequency: Count of documents within which a term appears
  - Usually divided by the total number of documents
- Inverse document frequency: N/DF
  - Higher DF = Lower weight, Lower DF = Higher weight
- TF*IDF, term count weighted by how “informative” that term is
- Additional processing
  - Log-transform
  - Smoothing

TF-IDF

Word	df	idf
Romeo	1	1.57
salad	2	1.27
Falstaff	4	0.967
forest	12	0.489
battle	21	0.246
wit	34	0.037
fool	36	0.012
good	37	0
sweet	37	0

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.074	0	0.22	0.28
good	0	0	0	0
fool	0.019	0.021	0.0036	0.0083
wit	0.049	0.044	0.018	0.022

Figure 6.8 A tf-idf weighted term-document matrix for four words in four Shakespeare plays, using the counts in Fig. 6.2. For example the 0.049 value for *wit* in *As You Like It* is the product of $\text{tf} = \log_{10}(20+1) = 1.322$ and $\text{idf} = .037$. Note that the idf weighting has eliminated the importance of the ubiquitous word *good* and vastly reduced the impact of the almost-ubiquitous word *fool*.

https://web.stanford.edu/~jurafsky/slp3/6.pdf

Implementation in sklearn

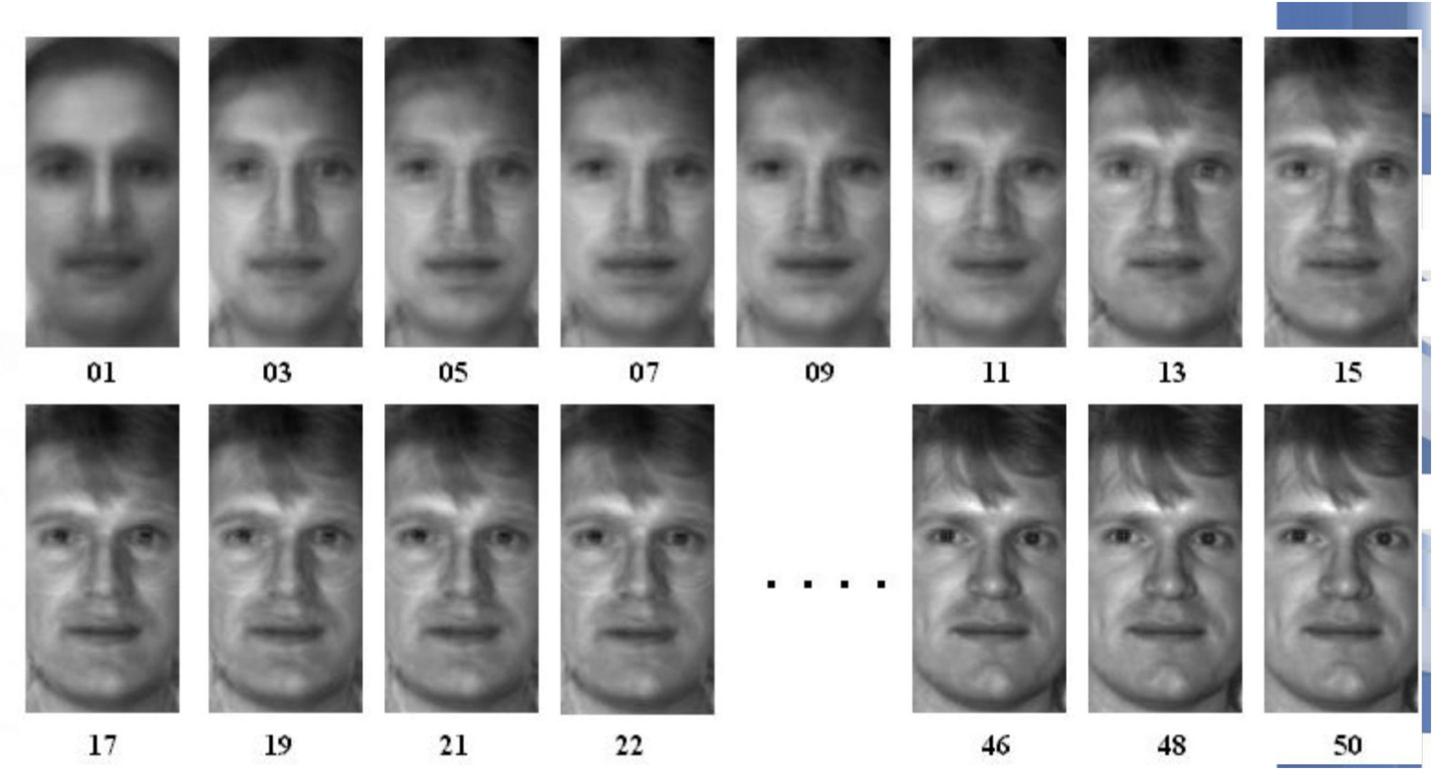
Topic models

- “Topic models are algorithms for discovering the main themes that pervade a large and otherwise unstructured collection of documents” (Blei 2012)
- Document = f(topics), Topics = g(words)
  - Typically number of topics << size of vocabulary
  - Want to minimize the information lost by representing in this way
- Typically for unsupervised problems
  - Creating topics when you don’t already have them labelled

Extracting axes of variation in data

https://jakevdp.github.io/PythonDataScienceHandbook/05.09-principal-component-analysis.html

Application in image processing

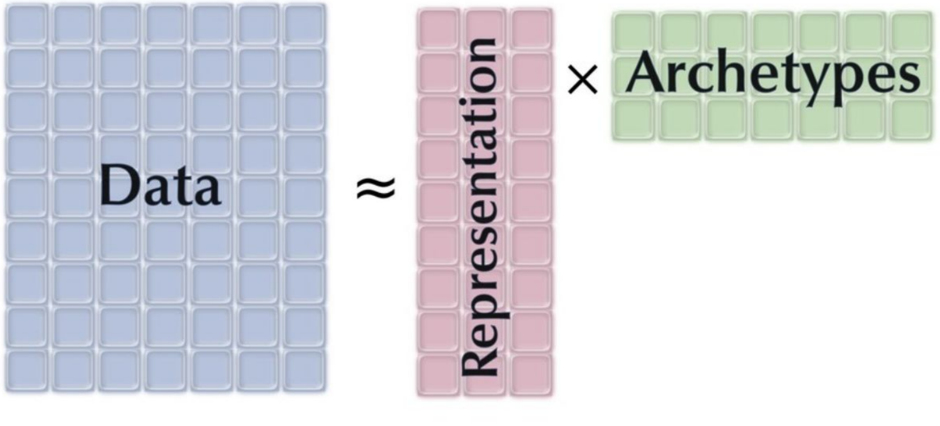


https://www.cec.uchile.cl/~jruizd/faces/reconstruction/rec.htm

Distilling text vectors with matrix factorization

- Matrix factorization: Decomposing a matrix into archetypes and values
- In NLP: Extracting “latent” structure of the association between terms and documents
- The number of archetypes is typically lower than the number of features

Visual of factorization



X U V

Minimize

$$\sum_{i=1}^{N} \sum_{j=1}^{D} \textbf{Loss}\left(X_{ij}, (UV)_{ij}\right)$$

Leland McInnes: Bluffer’s Guide to Matrix Factorization
https://www.youtube.com/watch?v=9iol3Lk6kyU
https://speakerdeck.com/lmcinnes/a-guide-to-dimension-reduction

LSI vs Non-negative Matrix Factorization (NMF)

LSI

Minimize

$$\sum_{i=1}^{N}\sum_{j=1}^{D}\left(X_{ij}-(UV)_{ij}\right)^2$$

with no constraints

NMF

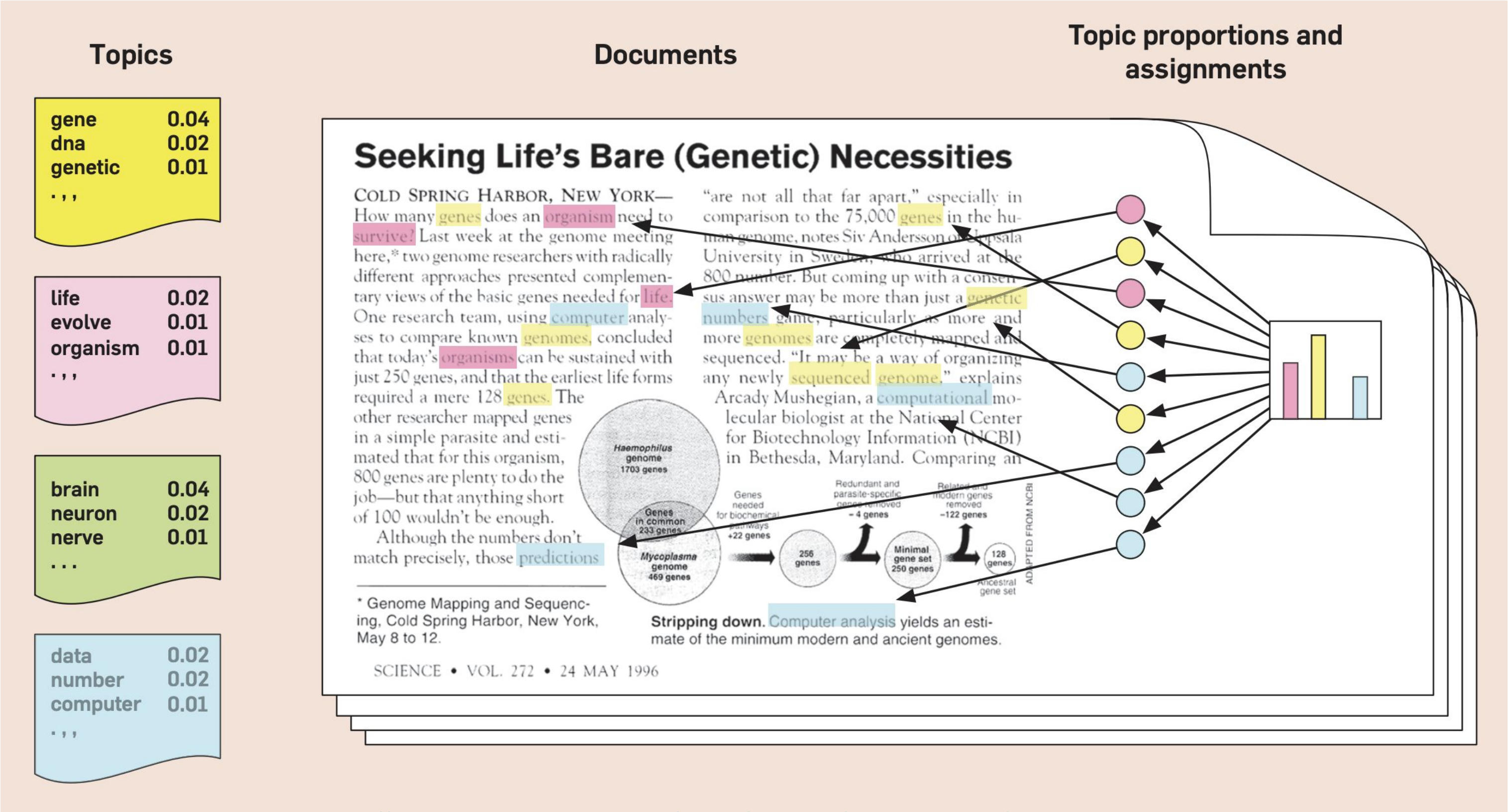
Minimize

$$\sum_{i=1}^{N}\sum_{j=1}^{D}\left(X_{ij}-(UV)_{ij}\right)^2$$

Subject to

$U_{ij} \geq 0$ **and** $V_{ij} \geq 0$

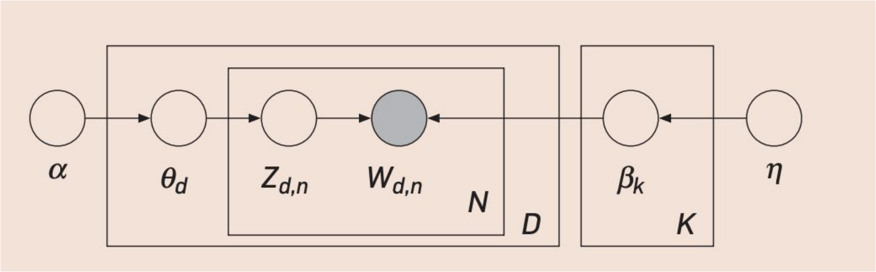
Latent Dirichlet Allocation (LDA)



http://www.cs.columbia.edu/~blei/papers/Blei2012.pdf

The LDA generative process

- α/η = parameters governing the distributions from which theta+beta are drawn
- K = topics,
- D = docs
- N = words
- $\boldsymbol{\theta}$ = document's distribution over topics
- β = word distribution over topics
- Z = the topic assignment of word n in document d



http://www.cs.columbia.edu/~blei/papers/Blei2012.pdf

NMF + LDA implementation