Latent Variables and Natural Language Processing

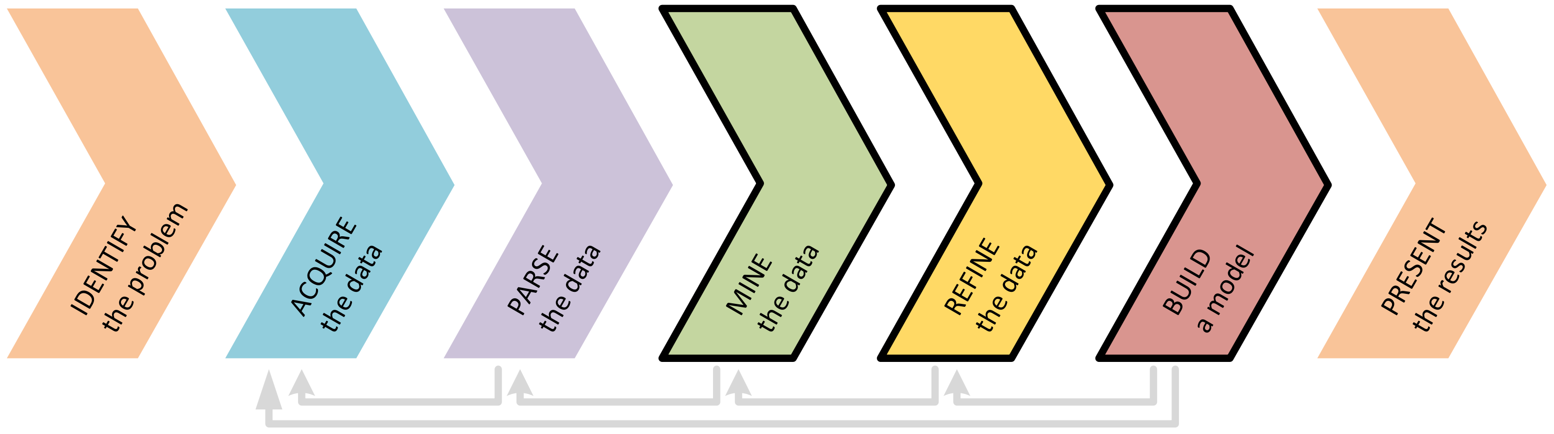
Ivan Corneillet

Data Scientist

GENERAL ASSEMBLY

Where Natural Language Processing fit in the course

Unit 1 – Research Design and Data Analysis	*Research Design*	*Data Visualization in Pandas*	*Statistics*	*Exploratory Data Analysis in Pandas*
Unit 2 – Foundations of Modeling	Linear Regression	Classification Models	Evaluating Model Fit	Presenting Insights from Data Models
Unit 3 – Data Science in the Real World	*Decision Trees and Random Forests*	*Time Series Data*	*Natural Language Processing*	*Databases*



Learning Objectives

After this lesson, you should be able to:

- Understand what *latent variables* are
- Understand the uses of *latent variables* in language processing
- Use the *Word2Vec* and LDA algorithms of *genism*

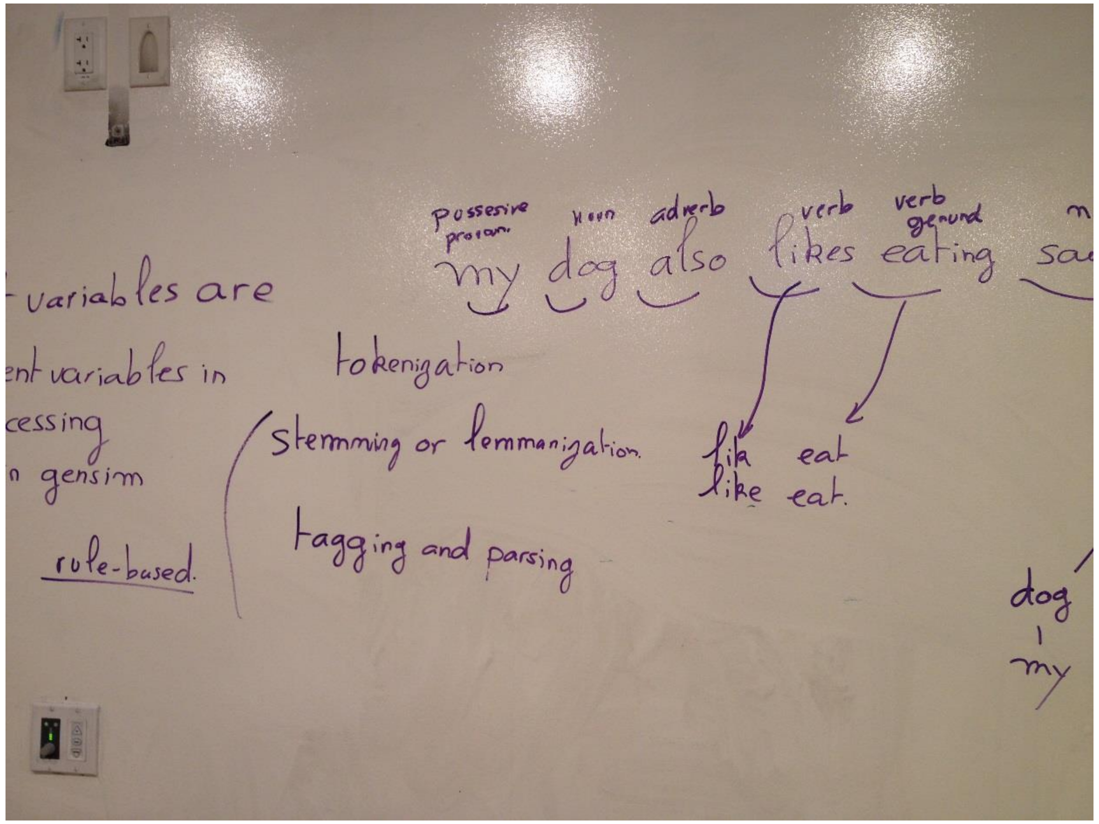
Outline

- Final Project 2 (due today)
- Review
- Latent Variable Models
- Dimensionality Reduction
- Dimensionality Reduction in Text Representation
  - Principal Component Analysis (PCA) and Latent Semantic Indexing (LSI)
- Mixture Models and Language Processing
  - Latent Dirichlet Allocation (LDA)
  - Word2Vec
- Independent Practice
  - Tweets and Word2Vec
  - Bonus – Capture your own tweets and redo your work on these
- Office hours in class for final projects
- Review
- Assigned
  - Final Projects 3 (due on 4/19), 4 (due on 4/26) , and 5 (due on 4/28)

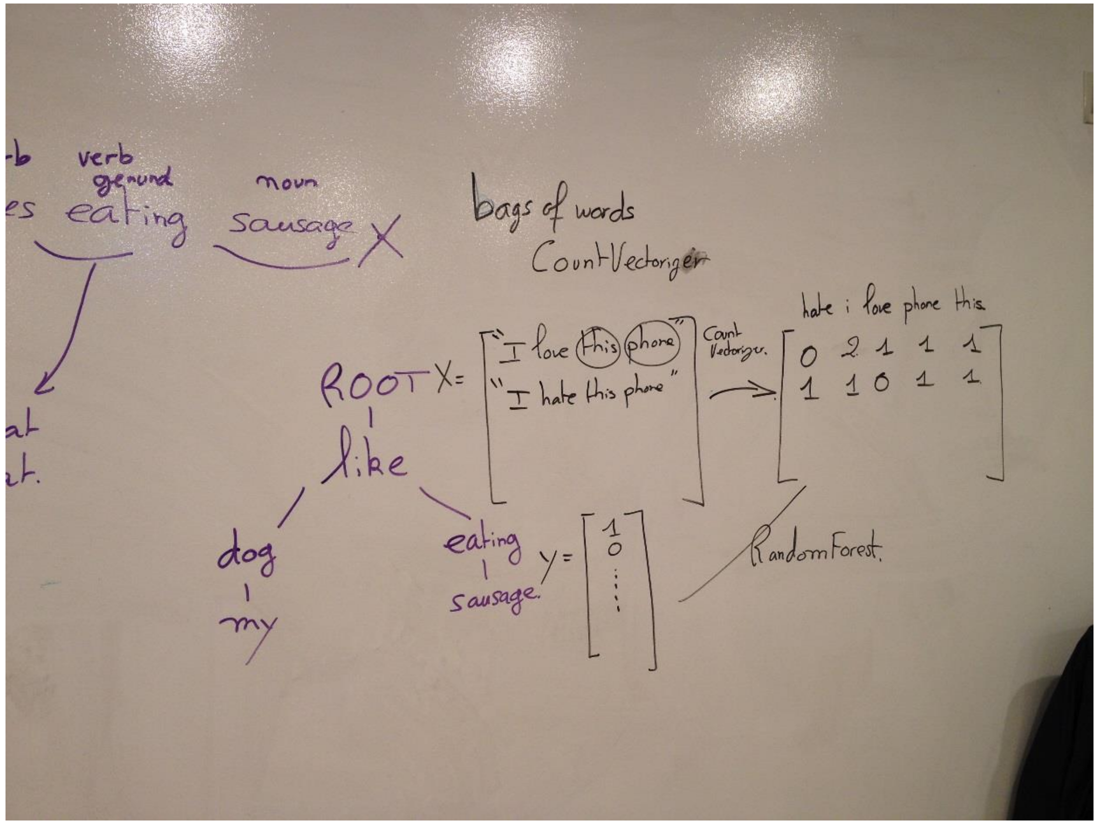
DS

Review

Review



variables are
ent variables in
cessing
n gensim
rule-based
tokenization
stemming or lemmanization
tagging and parsing
possesive pronoun
noun
adverb
verb
verb genund
my dog also likes eating sa
lik
like
eat
eat.
dog
my



verb genund
noun
es eating sausage
at
at.
bags of words
CountVectorizer
ROOT
like
dog
my
eating
sausage
X=
"I love this phone"
"I hate this phone"
Count Vectorizer
hate i love phone this
0 2 1 1 1
1 1 0 1 1
y=
1
0
RandomForest.

DS

Pre-Work

Pre-Work

Before the next lesson, you should already be able to:

- Install *gensim* with `pip install gensim`
- Recall and apply *unsupervised learning* techniques
- Recall probability distributions, specifically discrete multinomial distributions
- Recall NLP essentials, including experience with *spacy*

DS

Latent Variable Models

Latent Variable Models

- *Latent variable models* are different from the *traditional NLP models* in that they try to understand language based on **how** the words are used
  - E.g., instead of learning that ‘bad’ and ‘badly’ are related because they share the same root, we’ll determine that they are related because they are often used in the same way often or near the same words
- We’ll use unsupervised learning techniques to discover patterns or structure in the text to extract information

Latent Variable Models (cont.)

Traditional NLP Models	**Latent Variable Models**
❑ Focused on theoretical understanding of language	❑ Focused on how the language is actually used in practice
❑ Tries to learn the rules of a particular language	❑ Infers meaning from how words are used together
❑ Preprogrammed set of rules	❑ Uses unsupervised learning to discover patterns or structure

Latent Variable Models (cont.)

Traditional NLP Models	**Latent Variable Models**
❑ 'bad' and 'badly' are related because they share a common root	❑ 'bad' and 'badly' are related because they are used the same way or near the same words
❑ 'Python' and 'C++' are both programming languages because they are often a noun preceded by the verb 'program' or 'code'	❑ 'Python' and 'C++' are both programming languages because they are often used in the same context

Latent Variable Models (cont.)

- *Latent variable models* are models in which we assume the data we are observing has some **hidden**, **underlying structure** that we can't see, but which we'd like to learn
- These hidden, underlying structures are the *latent* (i.e., hidden) variables we want our model to understand

Text processing is a common application of latent variables

- While language (in the classical sense) is defined by a set of pre-structured grammar rules and vocab, we often break those rules and create new words (e.g., ‘selfie’)
- Instead of attempting to train our model on the rules of proper grammar, we’ll ignore grammar and seek to uncover alternate hidden structures

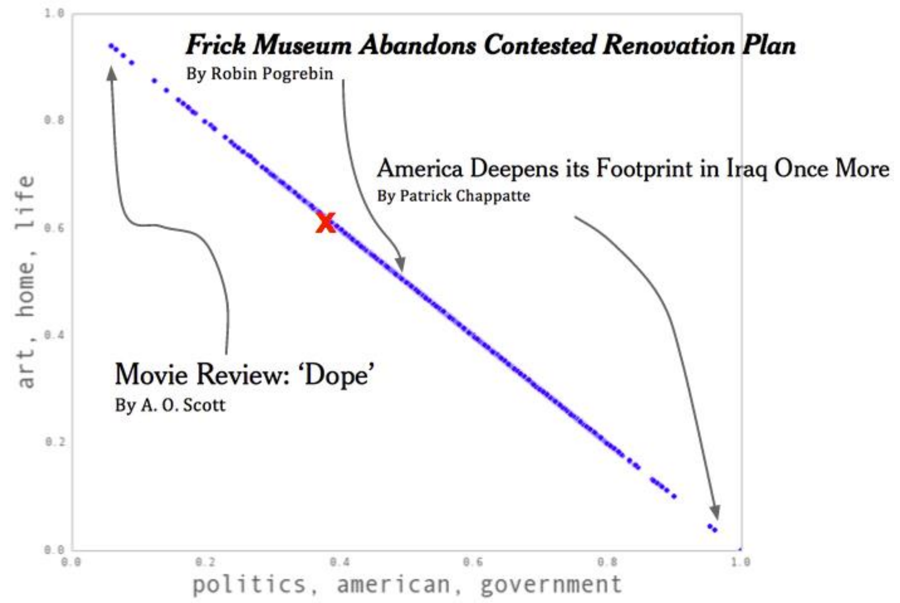
Topic modeling is another common application of latent variables

- I.e., recommending news articles or mining large troves of data to find commonalities

The New York Times (http://www.nytimes.com/#recommendations) uses topic modeling to map their articles to a latent space of topics using the articles content

EXAMPLE

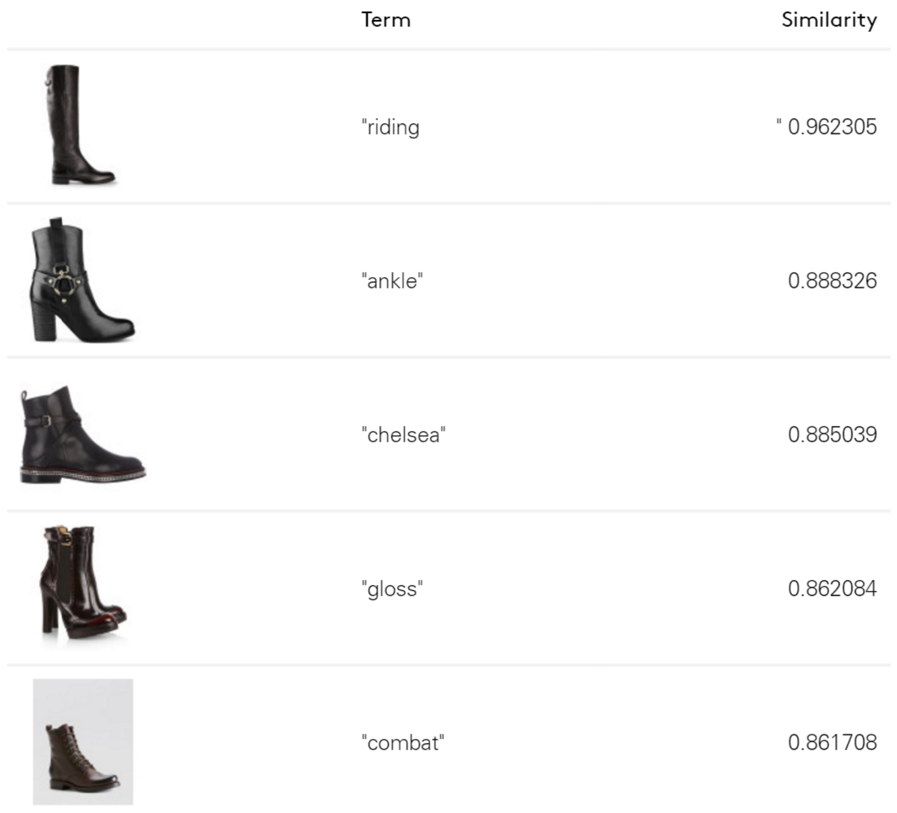
- 2 latent topics, “Art” and “Politics”
- Politics–Art space
  - “America Deepens its Footprint in Iraq Once More”, as 100% Politics
  - Film review by A.O. Scott as 100% Art
  - “Frick Museum Abandons Contested Renovation Plan” as 50% Politics, 50% Art
- Reader (x) prefers to read about Art 60% of the time and Politics 40% of the time
  - The New York Times will recommend articles that are closest to them in the space



Lyst (https://www.lyst.com/), an online fashion retailer, uses latent representations of clothing descriptions to find similar clothing

EXAMPLE

- Looking for items most similar to “boot” mostly return types of boots, as well as some close synonyms to “boot”, like “boots”, “bootie”, and “booties”

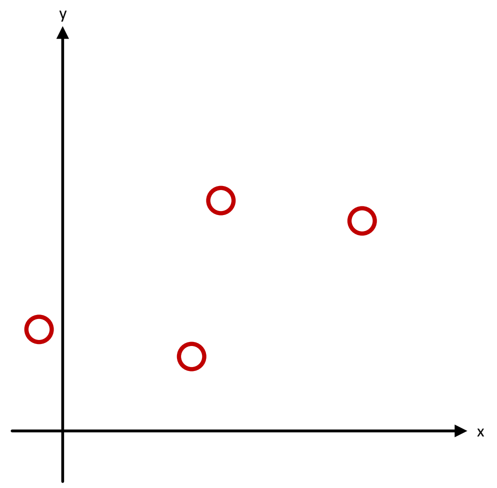


	Term	Similarity
	"riding	" 0.962305
	"ankle"	0.888326
	"chelsea"	0.885039
	"gloss"	0.862084
	"combat"	0.861708

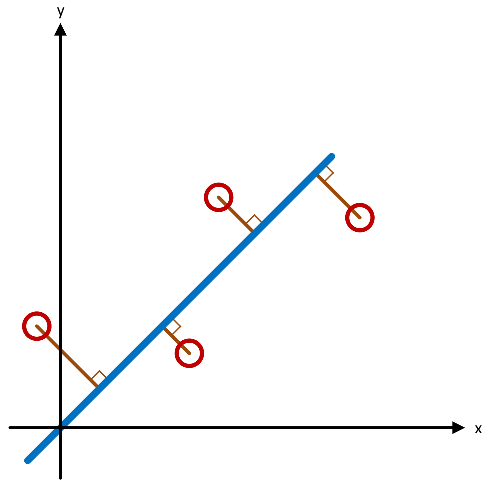
DS

Dimensionality Reduction

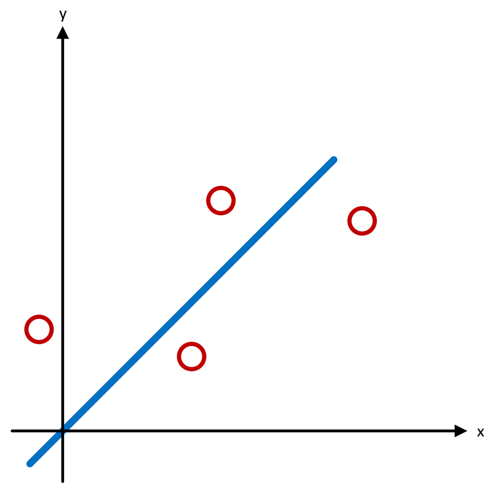
How to reduce dimensionality (2D down to 1D) while keeping as much information as possible?



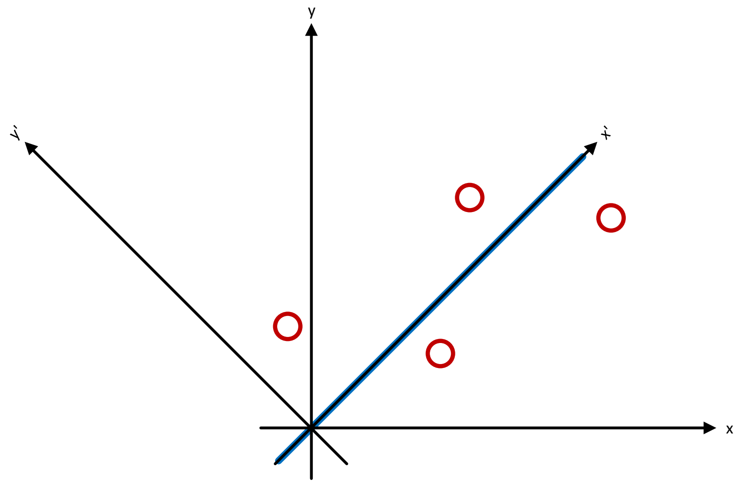
Let’s fit a “best” line



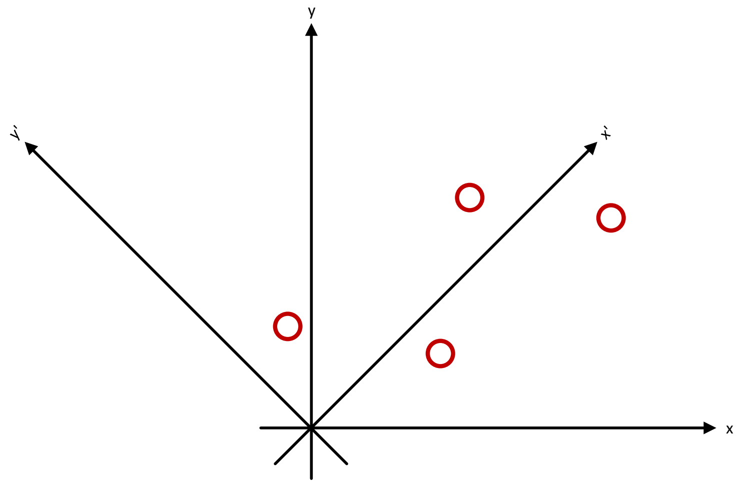
Let’s fit a “best” line (cont.)



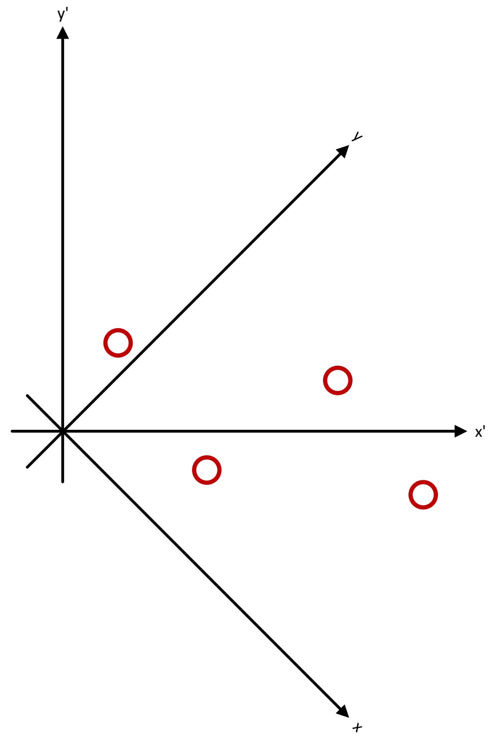
This “best” line define a new coordinate system (x’ along the line and y’ orthogonal to x’)



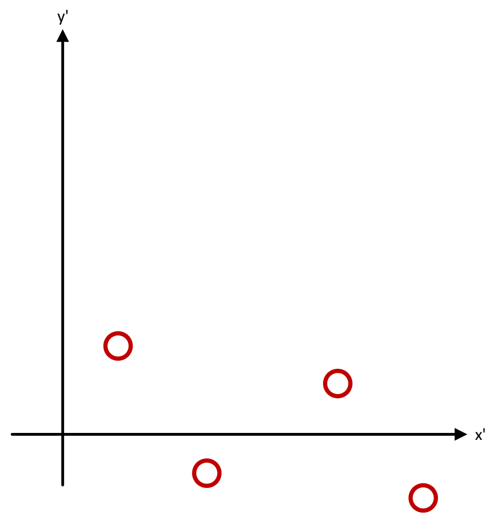
This "best" line define a new coordinate system (x' along the line and y' orthogonal to x') (cont.)



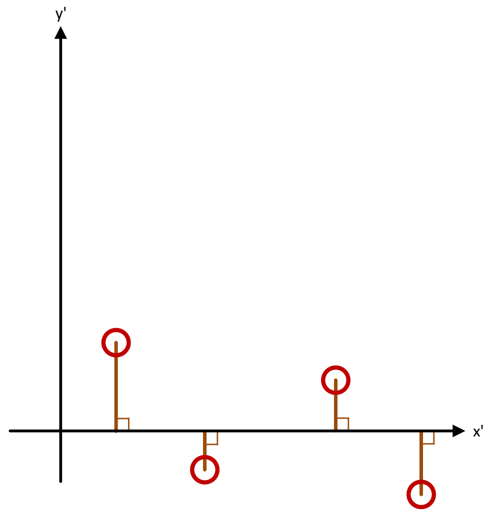
Let’s consider the new coordinate system (x’/y’)



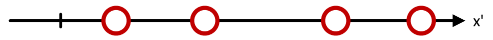
And let’s remove the x/y coordinate system



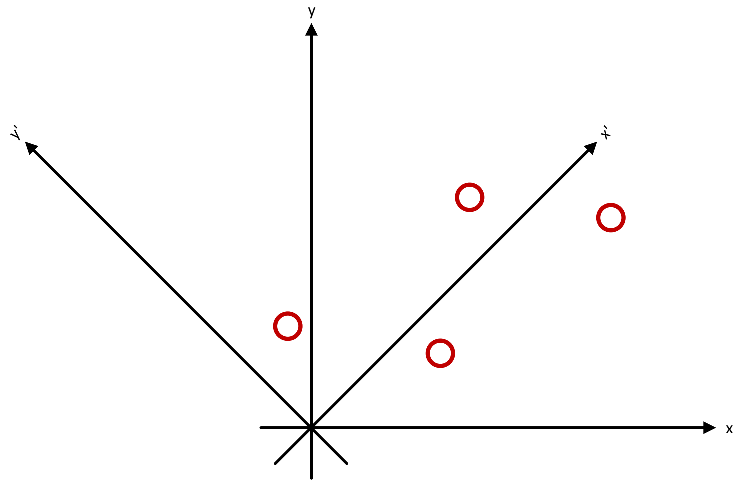
Let’s now project the data points into x’



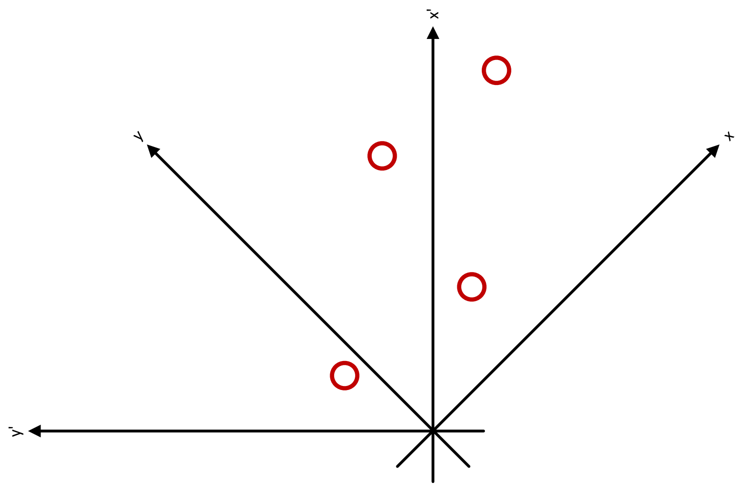
Let’s now project the data points into x’ (cont.)



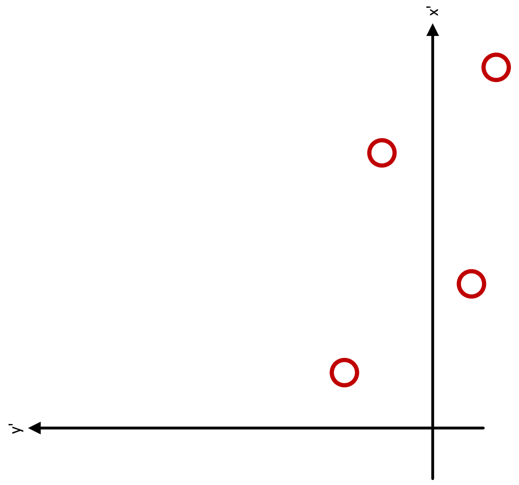
Let’s do the same work for y’



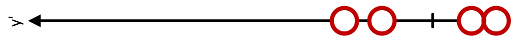
Let’s do the same work for y’ (cont.)



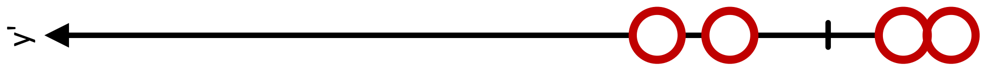
Let’s do the same work for the y’ axis (cont.)

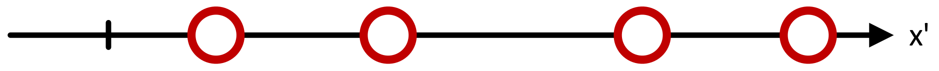


Let’s do the same work for y’ (cont.)

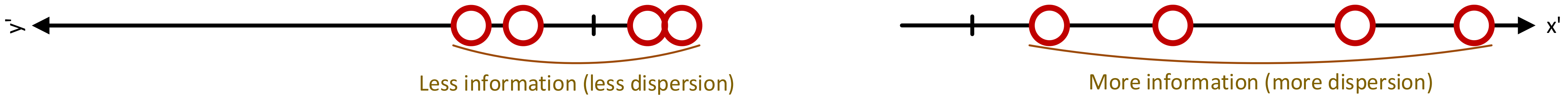


How much information have we “lost” when projecting the data points to x’/y’?

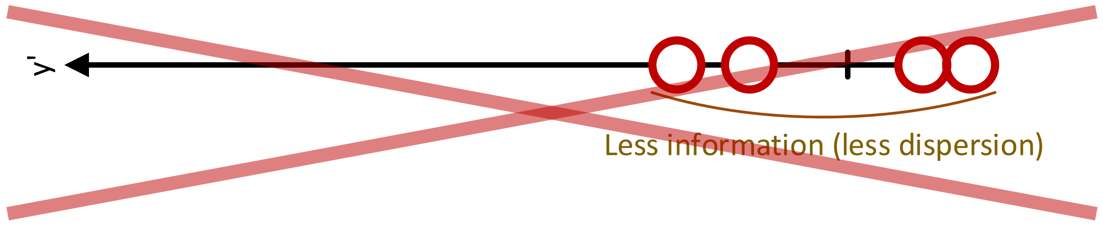


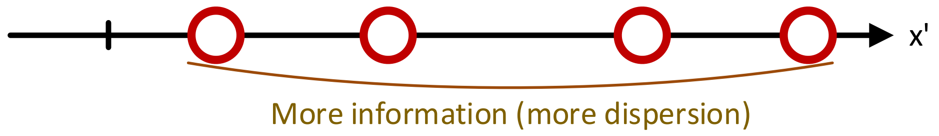


Which axis should we take if we need to forego the other?

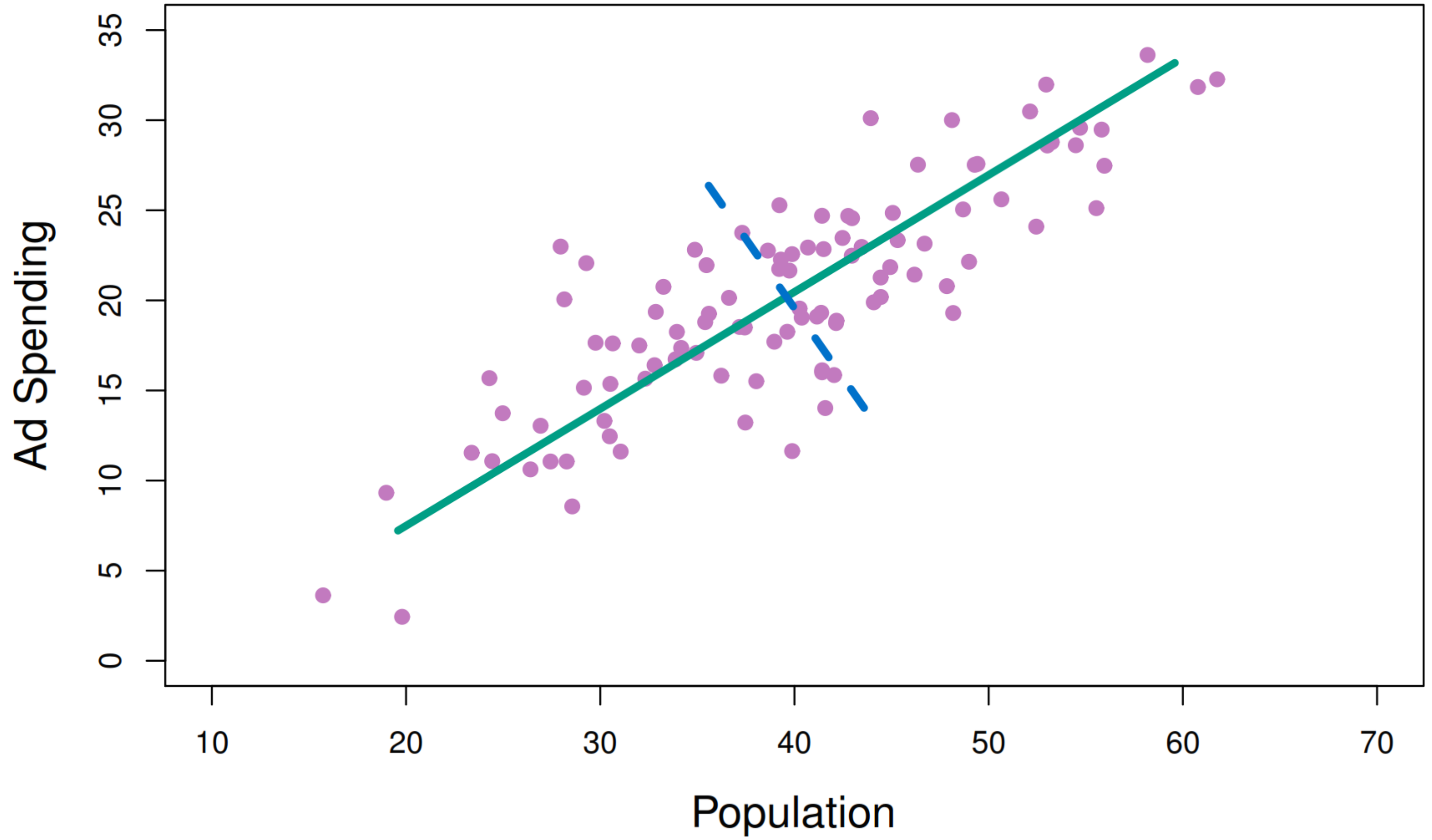

y'
Less information (less dispersion)
More information (more dispersion)
x'

The one with the “most” information…



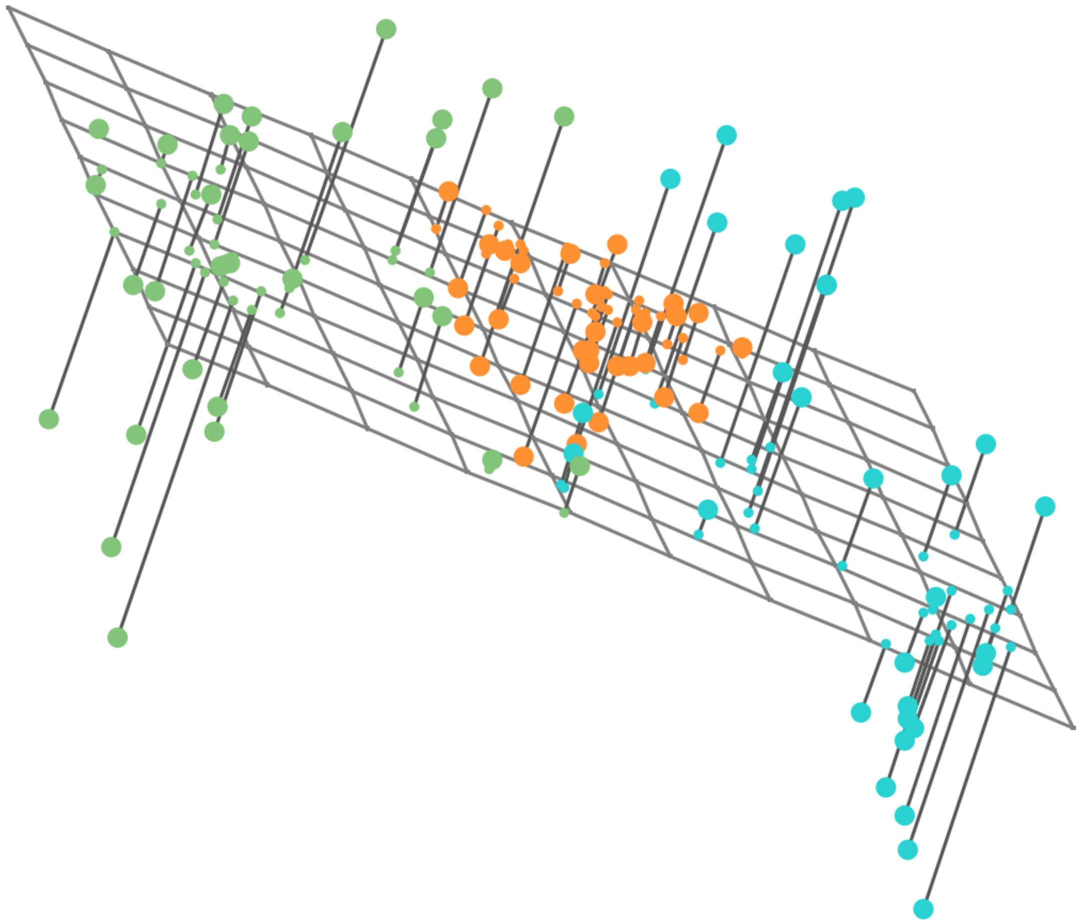


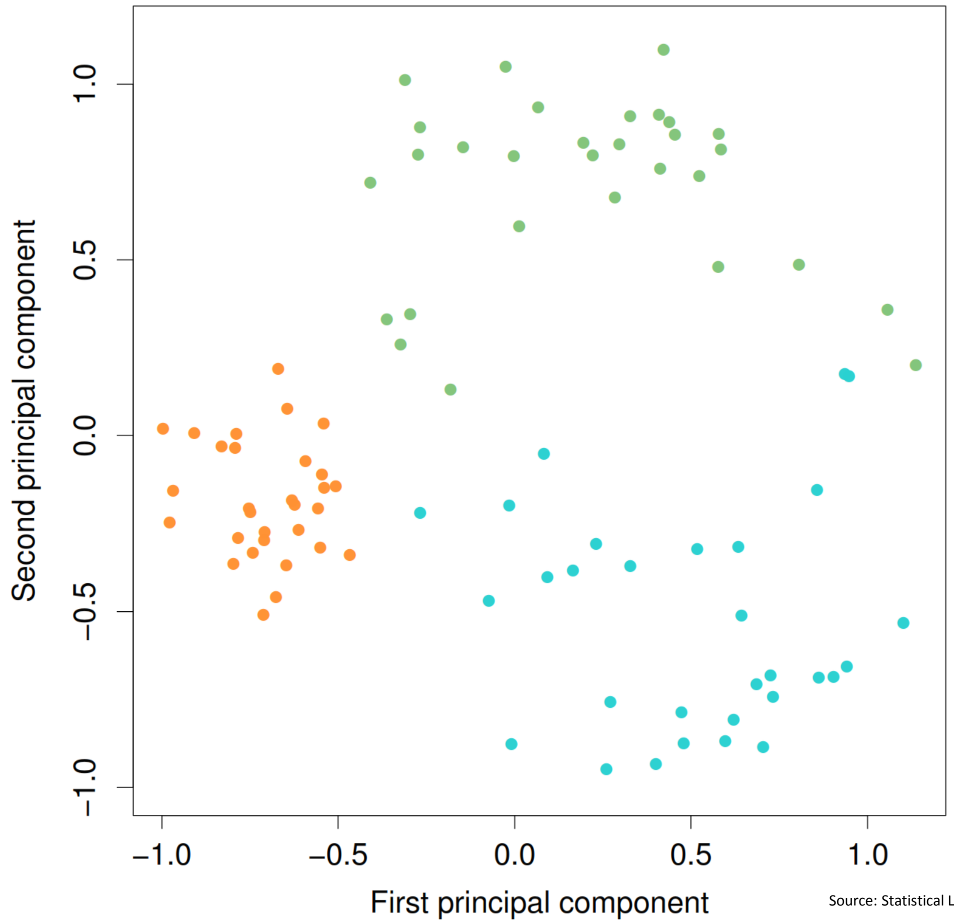
More examples of dimensionality reduction



Source: Statistical Learning

More examples of dimensionality reduction (cont.)





Source: Statistical Learning

Dimensionality Reduction in Text Representation

Dimensionality Reduction in Text Representation

- Our previous representation of a set of text documents (articles) for classification was a matrix with one row per document and one column per word (or n-gram)
- While this sums up most of the information, it does drop a few things, mostly structure and order

Additionally, many of the columns may be correlated

- E.g., an article that contains the word “IPO” is likely to contain the word “stock” or “NASDAQ”
  - These columns are repetitive and likely to represent the same concept or idea
  - For classification, we may only care that there are finance-related words
- One way to deal with this is to perform dimensionality reduction, where we first identify the correlated columns and the replace them with a column that represents the concept they have in common
  - E.g., we could replace the ‘`IPO`’, ‘`stock`’, and ‘`NASDAQ`’ columns with a single column ‘`HasFinancialWords`’ column

Principal Component Analysis (PCA)

- The techniques vary in how they define correlation and how much of the relationship between the original and new columns you need to save
- One of the most common is Principal Component Analysis (PCA)
- PCA helps reduce the feature space into fewer dimensions
- PCA, when applied to text data, is sometimes known as Latent Semantic Indexing (LSI)

Mixture Models and Language Processing

Mixture models take the dimensionality reduction concept further to generate more structure around document

- Mixture models assume that each document is some mixture of topics
    - E.g., an article may be mostly science but may contain some business information
- The latent structure we want to uncover are the topics (or concepts) that generate that text
- Mixture models create clusters of common words and generate probability distributions to explicitly state how related words are

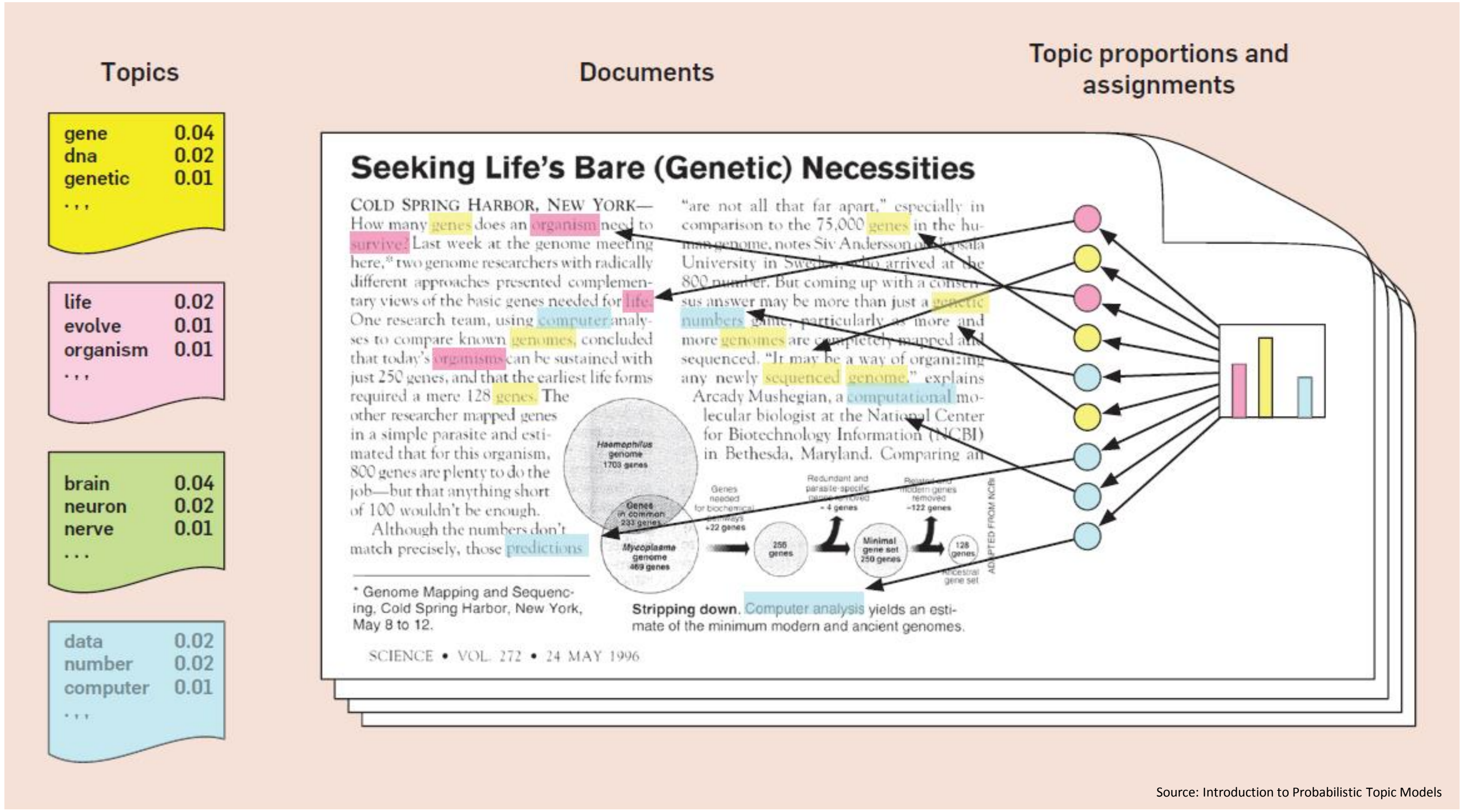
DS

Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) is an unsupervised mixture model for latent variable NLP

- LDA attempts to learn two things:
    - The word distribution of each topic
    - The topic distribution of each document

Mixture Models (cont.)


Topics
gene 0.04
dna 0.02
genetic 0.01
...
life 0.02
evolve 0.01
organism 0.01
...
brain 0.04
neuron 0.02
nerve 0.01
...
data 0.02
number 0.02
computer 0.01
...
Documents
Topic proportions and assignments
Haemophilus genome 1703 genes
Genes in common 233 genes
Mycoplasma genome 469 genes
Genes needed for biochemical pathways +22 genes
256 genes
Redundant and parasite-specific genes removed −4 genes
Minimal gene set 250 genes
Related and modern genes removed −122 genes
128 genes
Ancestral gene set
ADAPTED FROM NCBI

Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK— How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

Although the numbers don't match precisely, those predictions "are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game, particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an

Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.

* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

SCIENCE • VOL. 272 • 24 MAY 1996

Source: Introduction to Probabilistic Topic Models

The word distribution of each topic

- The word distribution is a multinomial distribution of each topic representing what words are most likely from that topic
- For each word and topic pair, we learn some probability: $P(word \mid topic)$

- E.g., we have three topics: sports, business, and science
- For each topic, we uncover the most likely words to come from them
  - `sports→[football: .3, basketball: .2, baseball: .2, touchdown: .02, …, genetics: .0001]`
  - `science→[genetics: .2, drug: .2, …, baseball: .0001]`
  - `business→[stocks: .1, ipo: .08, …, baseball: .0001]`

The topic distribution of each document

- The topic distribution is a multinomial distribution for each document representing what topics are most likely to appear in that document
- For each topic and document pair, we learn some probability: $P(topic \mid document)$

- E.g., for all our of sample documents, we have a distribution over sports, science, and business
  - `ESPN→[sports: .8, business: .2, science: .0]`
  - `Bloomberg→[business: .7, science: .2, sports: .1]`

Mixture Models (cont.)

- Topic models are useful for organizing a collection of documents and uncovering the main underlying concepts
- There are many variants that attempt to add even more structure:
  - Supervised topic models guide the process with pre-decided topics
  - Position-dependent topic models ignore which words occur in which document and instead focus on where they occur
  - Variable number topic models test different numbers of topics to find the best model

DS

Demo – LDA with *gensim*

DS

Word2Vec

Word2Vec is another unsupervised model for latent variable NLP

- Word2Vec model creates word vectors, multidimensional representations of words
- This is similar to having a distribution of concepts or topics that the word may come from

- E.g., `assembly`→`[.12315, .23425, .89745324, .235234, .234234, …]`

Documents as features of a specific word

- If we transpose our usual document-word matrix, instead of talking about words as being features of a document, we can talk about documents as being features of a specific word
- In other words, how do we define or characterize a single word?
  - We can do so by defining its dictionary definition
  - Or we can enumerate all of the ways we might use it

E.g., Paris

- Given the word 'Paris', we have many contexts or uses we may find it in:
  - `['_ is the capital of', '_, France', 'the capital city _', 'the restaurant in _', …]`
- There are also a bunch of contexts we don't expect to find it in:
  - `['can I have a _', 'there's too much _ on this', …]`
- We could make a feature or column for each of these contexts and even represent 'Paris' in a sparse feature with all possible contexts

E.g., Paris (cont.)

- In fact, the first few examples represent the same concept:
  - Paris is a city like thing, so it contains shops and restaurants
  - Paris is a capital city
- We would want to use dimensionality reduction to find a *few* concepts per word instead of *all* possible contexts

Word2Vec (cont.)

- With LDA, we could do this by identifying the topics a word was most likely to come from
- With Word2Vec, we will replace the overlapping contexts by some concept that represents them
- Like other techniques, our goal is to identify correlated columns and replace them with a new column that represents those replaced columns
- We would replace the `['_ is a city', '_ is a capital', 'I flew into _ today']` columns by a single column, ‘`IsACity`’

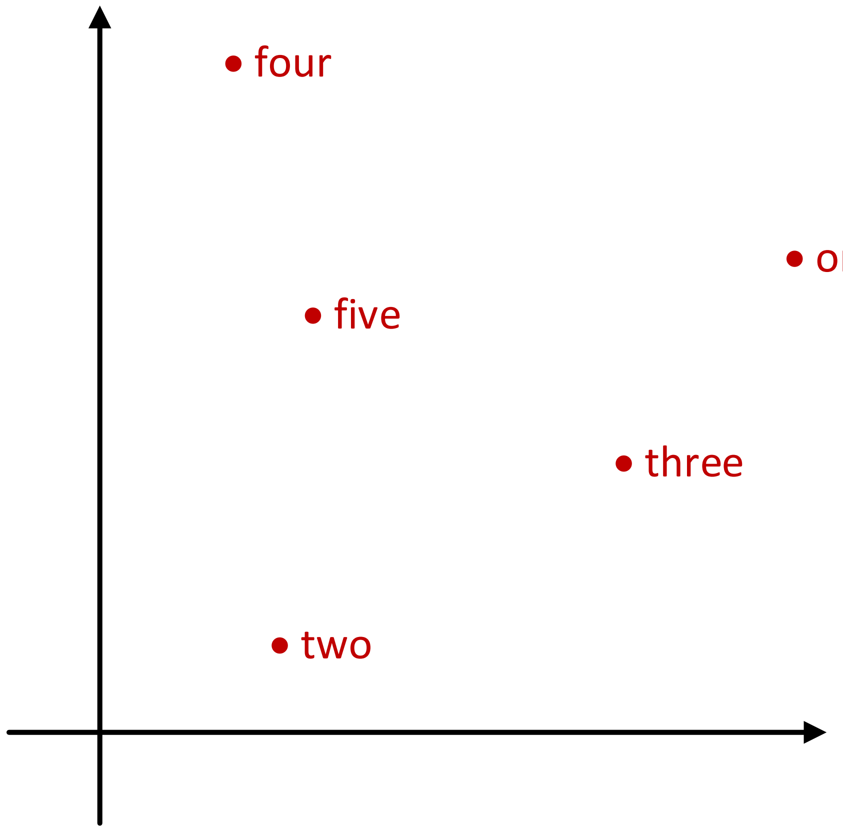
Word2Vec (cont.)

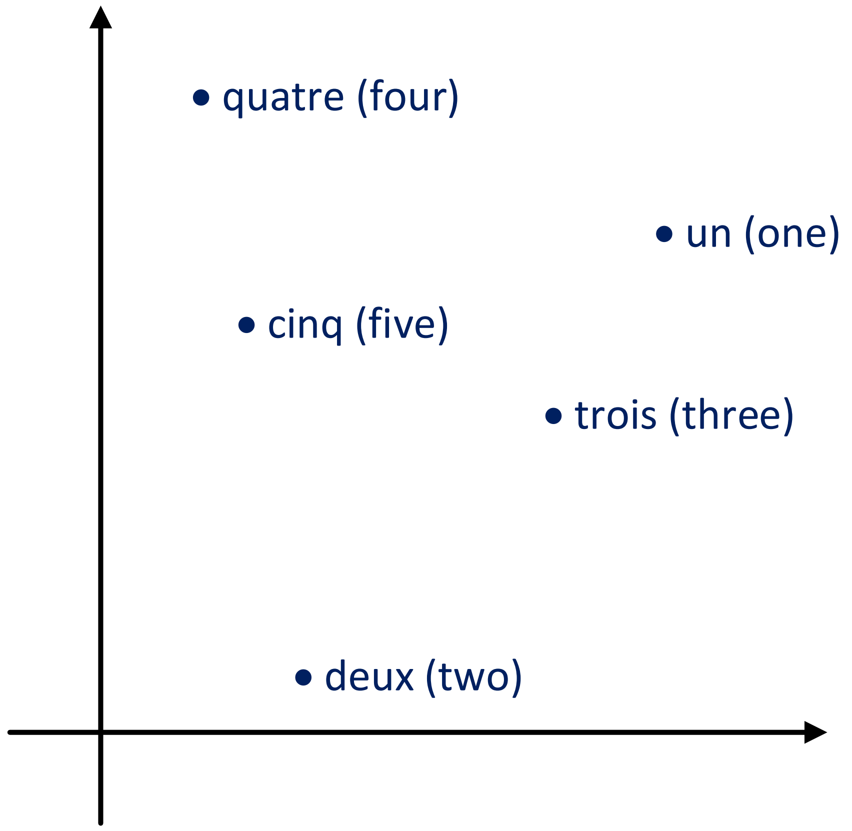
- With a trained model, Word2Vec can be used for many tasks
- A commonly used feature of Word2Vec is being able to ask what words are similar to each other

- For example, if you ask for words similar to ‘france’, you would get:

```
spain 0.678515
belgium 0.665923
netherlands 0.652428
italy 0.633130
switzerland 0.622323
luxembourg 0.610033
portugal 0.577154
```

Using data for other languages, Word2Vec could also be used for translation





DS

Demo – Word2Vec with *gensim*

DS

Independent Practice

Independent Practice: Tweets and Word2Vec

EXERCISE

TASKS AND QUESTIONS

1. Build a *Word2Vec* model of the tweets (`tweets.txt` under `datasets`) we have collected using *gensim*
   a) First take the collection of tweets and tokenize them
      - Think about how this should be done
      - Should you only use upper-case or lower-case?
      - Should you remove punctuations or symbols?

   b) Then, build a *Word2Vec* model

Independent Practice: Tweets and Word2Vec (cont.)

EXERCISE

TASKS AND QUESTIONS (cont.)

c) Finally, test your *Word2Vec* model with a few similarity functions

- Find words similar to 'Syria'
- Find words similar to 'war'
- Find words similar to 'Iran'
- Find words similar to 'Verizon'
- Adjust the choices in (b) and (c) as necessary

DELIVERABLE

Answers to the above questions

Independent Practice: Bonus – Capture your own tweets and redo your work on these

EXERCISE

SETTING UP TWITTER API CREDENTIALS

1. Go to https://apps.twitter.com/
2. Sign in, and follow the instructions below:
3. Press “Create New App”
4. Fill in form (you can enter your website or any website in the 'website' field, e.g. http://google.com)
5. Press “Create application”
6. Press “Manage keys”
7. Press “Create access tokens”
8. Find and save the following:
    a) Consumer Key
    b) Consumer Secret
    c) Access Token Key
    d) Access Token Secret

Independent Practice: Bonus – Capture your own tweets and redo your work on these (cont.)

EXERCISE

RUNNING THE CODE

1. Install TwitterAPI: `pip install TwitterAPI`
2. Substitute the four values saved from 'Setting up Twitter API credentials' in twitter.py in

```
access_token_key = "<ENTER ACCESS TOKEN KEY>"
access_token_secret = "<ENTER ACCESS TOKEN SECRET>"
api_key = "<ENTER CONSUMER KEY>"
api_secret = "<ENTER CONSUMER SECRET>"
```

3. Run `python capture-tweets.py <topic>` to save tweets to file called `captured-tweets.txt` related to `<topic>`. E.g.: `python capture-tweets.py Google` or `python capture-tweets.py Iran`

DS

Review

Review

- Latent variable models attempt to uncover structure from text
- Dimensionality reduction is focused on replacing correlated columns
- Topic modeling (or LDA) uncovers the topics that are most common to each document and then the words most common to those topics
- Word2Vec builds a representation of a word from the way it was used originally
- Both techniques avoid learning grammar rules and instead rely on large datasets. They learn based on how the words are used, making them very flexible

DS

Q & A

Exit Ticket

Don't forget to fill out your exit ticket [here]

Sources

- “Building the Next New York Times Recommendation Engine” (http://open.blogs.nytimes.com/2015/08/11/building-the-next-new-york-times-recommendation-engine/)
- “Word Embeddings For Fashion” (http://developers.lyst.com/2014/11/11/word-embeddings-for-fashion/)