

# LATENT VARIABLES AND NATURAL LANGUAGE PROCESSING

*Jim Simpson*

*Data Scientist, Sotera*

---

**OPENING**

---

# LATENT VARIABLE MODELS

---

# LATENT VARIABLE MODELS

---

- This lesson will continue on natural language processing with an emphasis on *latent variables models*.
- Mining and Refining data is a key part of the data science workflow.
- In our last class, we saw many techniques for mining the data, including preprocessing, building linguistic rules to uncover patterns, and creating classifiers from unstructured data.
- In this class, we'll continue with methods to Refine our understanding of the text by attempting to uncover structure or organization in the text.

---

# LATENT VARIABLE MODELS

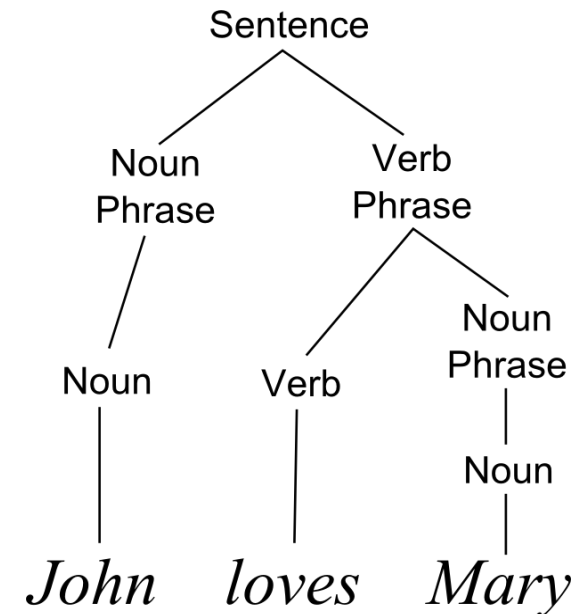
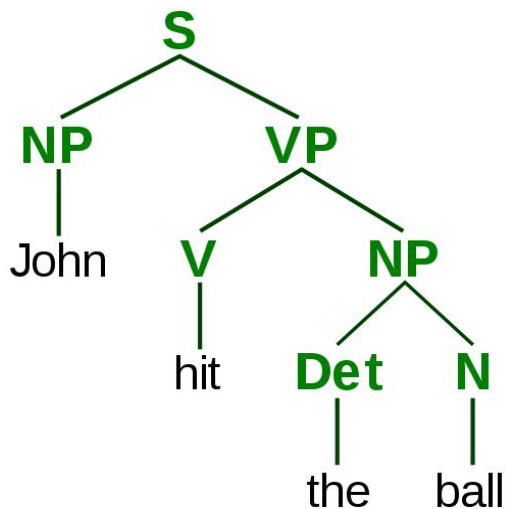
---

- Many advances in NLP are based on using data to learn rules of grammar and language. We saw these tools in our last class.
  - Tokenization
  - Stemming or lemmatization
  - Parsing and tagging
- Each of these are based on a classical or theoretical understanding of language.

# LATENT VARIABLE MODELS

- Tokenization:
  - John hit the ball → [John, hit, the, ball]
  - Where did you go → [Where, did, you, go]
- Stemming or lemmatization: shouted → shout, better → good

- Parsing and tagging:



---

# LATENT VARIABLE MODELS

---

- *Latent variable models* are different in that they try to understand language based on **how** the words are used.
- For example, 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* techniques (discovering patterns or structure) to extract the information.

---

# LATENT VARIABLE MODELS

---

## Traditional NLP Models

Focused on theoretical understanding of language

Tries to learn the rules of a particular language

Preprogrammed set of rules

## Latent Variable Models

Focused on how the language is actually used in practice

Infers meaning from how words are used together

Uses unsupervised learning to discover patterns or structure

---

# LATENT VARIABLE MODELS

---

## Traditional NLP Models

‘bad’ and ‘badly’ are related because they share a common root.

‘Python’ and ‘C++’ are both programming languages because they are often a noun preceded by the verb ‘program’ or ‘code’.

## Latent Variable Models

‘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 used in the same context.



## INTRODUCTION

---

# LATENT VARIABLE MODELS

---

# LATENT VARIABLE MODELS

---

- *Latent variable models* are models in which we assume the data we are observing has some **hidden, underlying structure** that we can't see, and 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.

---

# LATENT VARIABLE MODELS

---

- 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.

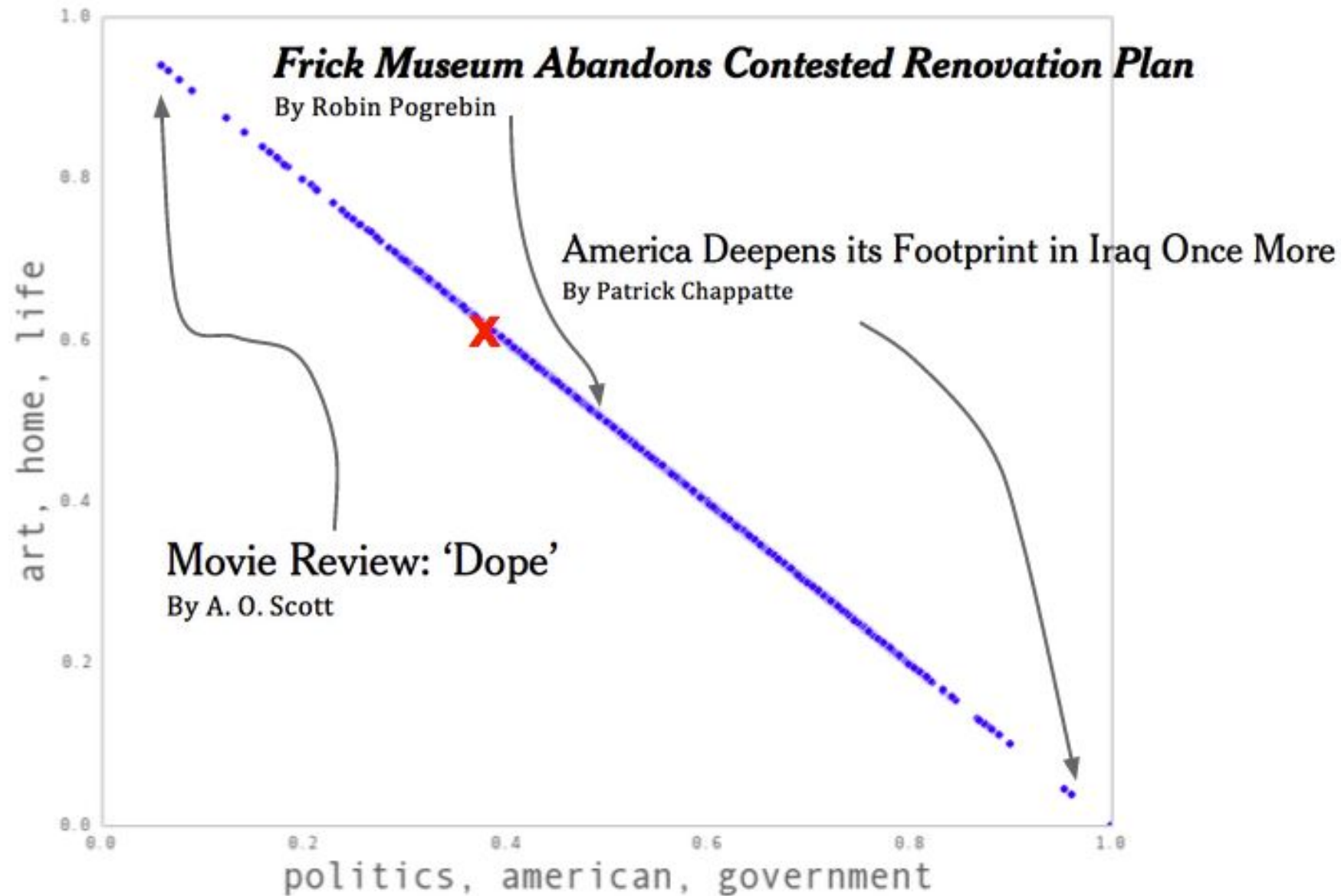
---

# LATENT VARIABLE MODELS

---






- Latent variable techniques are often used for recommending news articles or mining large troves of data to find commonalities.
- Topic modeling, a method we'll cover today, is used in [the NY times recommendation engine](#).
- The New York Times attempts to map their articles to a latent space of topics using the content of the article.

# LATENT VARIABLE MODELS



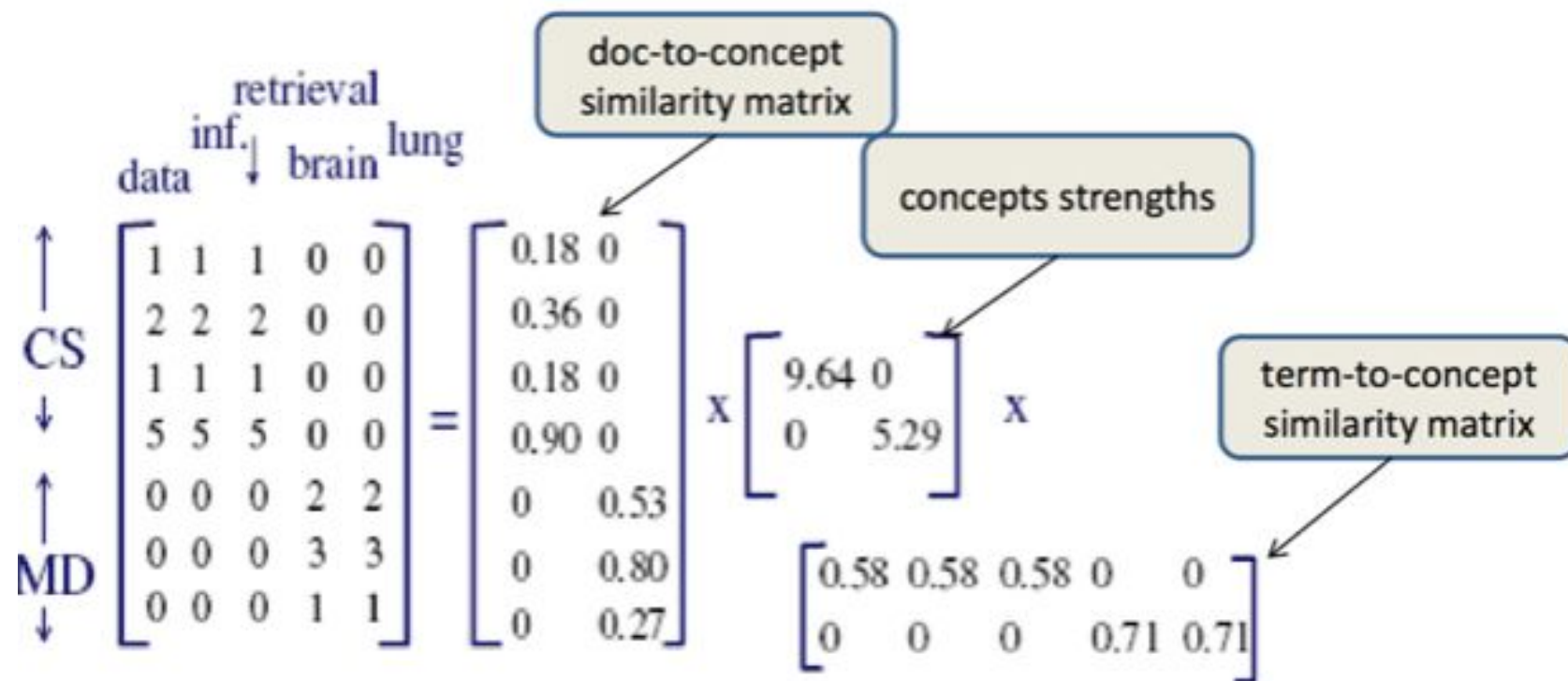
# LATENT VARIABLE MODELS

- [Lyst](#), an online fashion retailer, uses latent representations of clothing descriptions to find similar clothing.

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

# 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).



---

# **DIMENSIONALITY REDUCTION IN TEXT REPRESENTATION**

---

- 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.



---

## **DIMENSIONALITY REDUCTION IN TEXT REPRESENTATION**

---

- For example, an article that contains the word ‘IPO’ is likely to contain the word ‘stock’ or ‘NASDAQ’.
- Therefore, those columns are repetitive and likely to represent the same concept or idea.
- For classification, we may only care that there are finance-related words.

---

## DIMENSIONALITY REDUCTION IN TEXT REPRESENTATION

---

- One way to deal with this is through regularization - L1/Lasso regularization tends to remove repetitive features by bringing their learned coefficients to 0.
- Another is to perform *dimensionality reduction*, where we first identify the correlated columns and then replace them with a column that represents the concept they have in common.
- For instance, we could replace 'IPO', 'stocks', and 'NASDAQ' with a single column - 'HasFinancialWords' column.


---

# DIMENSIONALITY REDUCTION IN TEXT REPRESENTATION

---

- There are many techniques to do this automatically and most follow a very similar approach.
  - a. Identify correlated columns.
  - b. Replace them with a new column that encapsulates the others.

Doc #	Car	Truck	Van	Dog
6344	1	1	1	0
6345	0	1	1	1
6346	1	1	1	0



Doc #	Vehicle	Dog
6344	1	0
6345	1	1
6346	1	0

---

# DIMENSIONALITY REDUCTION IN TEXT REPRESENTATION

---

- The techniques vary in how they define correlation and how much of the relationship between the original and new columns you need to save.
- Dimensionality techniques can vary between *linear* and *non-linear*.

---

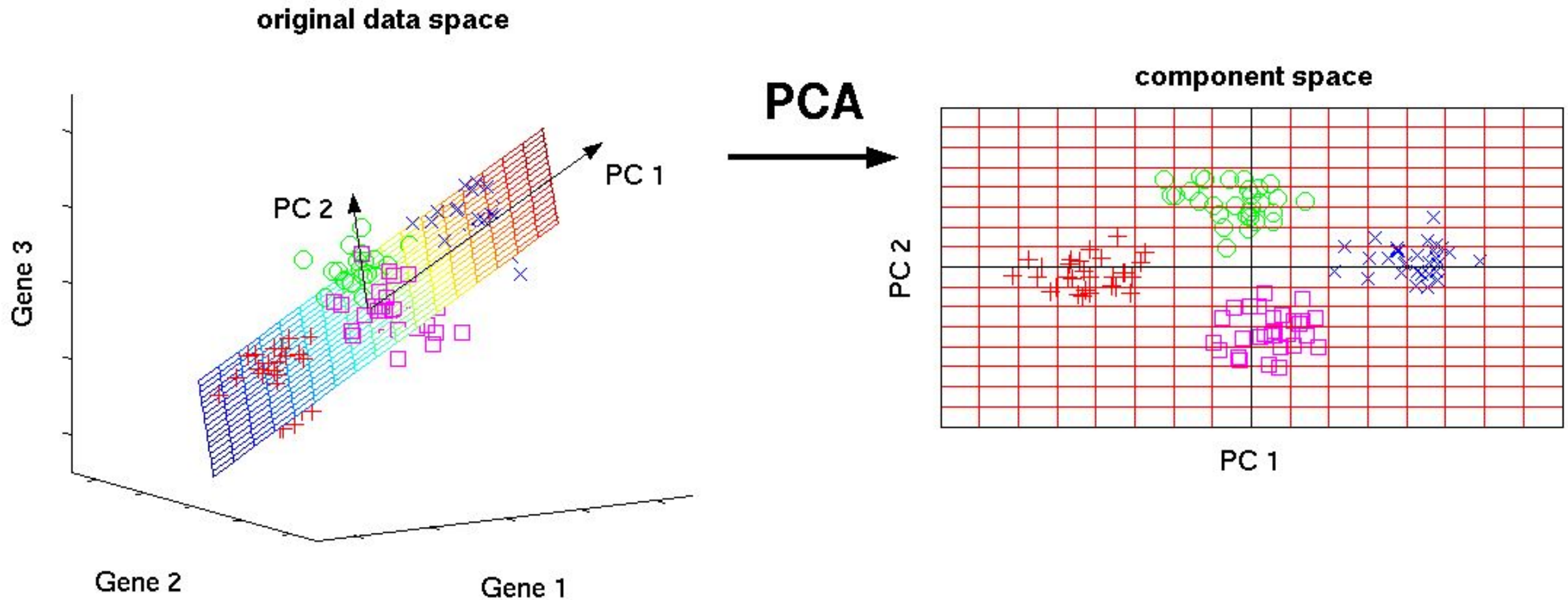
# DIMENSIONALITY REDUCTION IN TEXT REPRESENTATION

---

- There are many techniques build into scikit-learn.
- One of the most common is **Principal Component Analysis (PCA)**.
- PCA, when applied to text data, is sometimes known as **Latent Semantic Indexing (LSI)**.

# DIMENSIONALITY REDUCTION IN TEXT REPRESENTATION

- PCA helps reduce the feature space into fewer dimensions.



---

# MIXTURE MODELS AND LANGUAGE PROCESSING

---

- Mixture models (specifically **LDA** or **Latent Dirichlet Allocation**) take this concept further and generate more structure around the documents.
- Instead of just replacing correlated columns, we create clusters of common words and generate probability distributions to explicitly state how related words are.

---

# MIXTURE MODELS AND LANGUAGE PROCESSING

---

- This ‘model’ of text is assuming that each document is some *mixture* of topics.
- It 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 AND LANGUAGE PROCESSING

---

- *Latent Dirichlet Allocation* is a model that assumes this is the way text is generated and then attempts to learn two things:
  - a. The word distribution of each topic
  - b. The topic distribution of each document.

# MIXTURE MODELS AND LANGUAGE PROCESSING

## 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

### Seeking Life's Bare (Genetic) Necessities

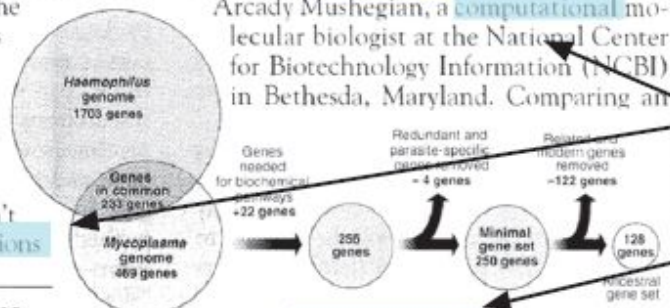
COLD SPRING HARBOR, NEW YORK—How many **genes** does an **organism** need to **survive**? Last week at the genome meeting here,\* two genome researchers with radically different approaches presented complementary views of the basic genes needed for **life**. One research team, using **computer** analyses 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**

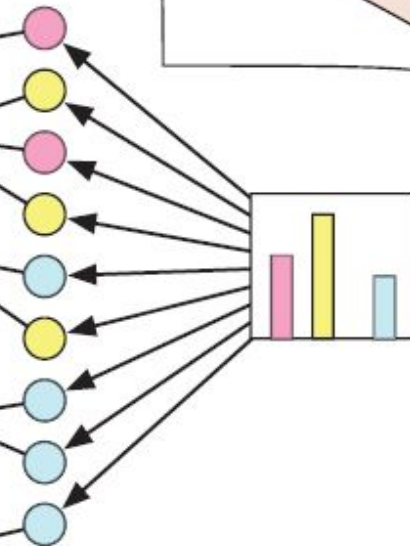
"are not all that far apart," especially in comparison to the 75,000 **genes** in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a **genetic numbers** game, particularly as more and more **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

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

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

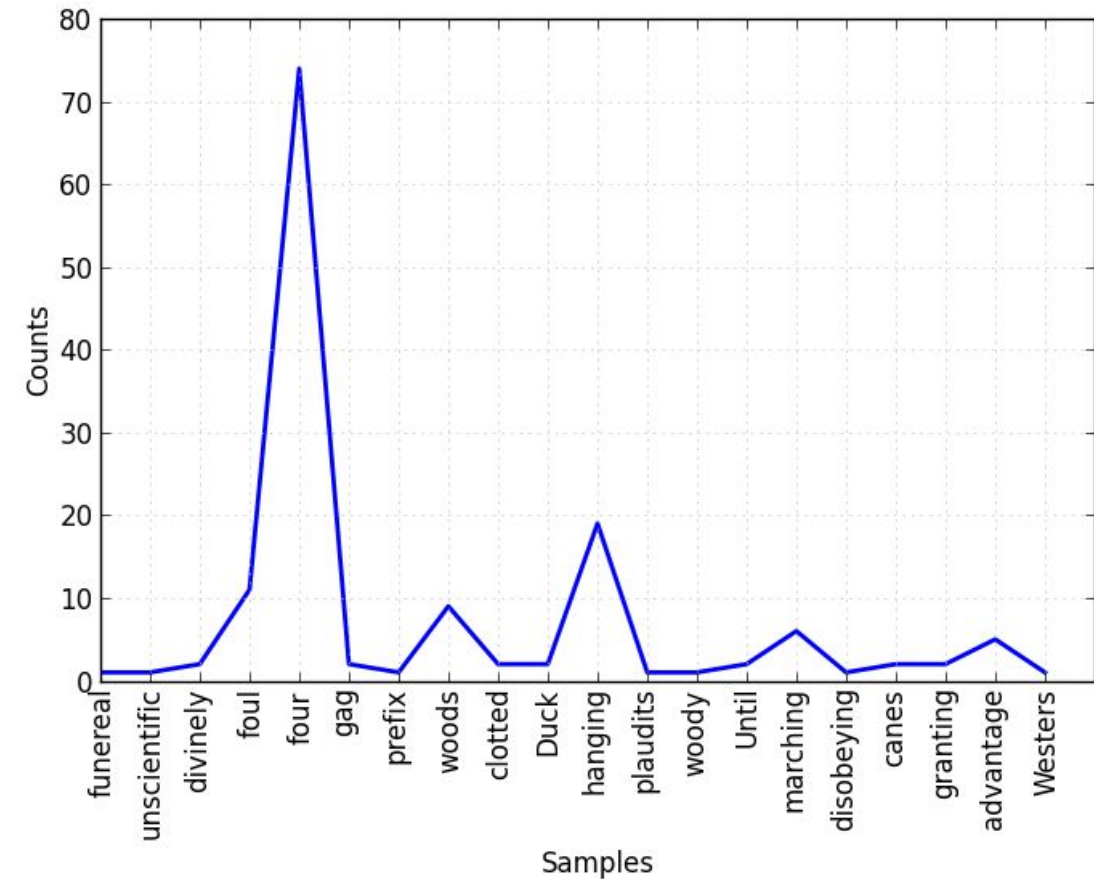


## Topic proportions and assignments



# MIXTURE MODELS AND LANGUAGE PROCESSING

- The *word distribution* is a multinomial distribution of each topic representing what words are most likely from that topic.



---

# MIXTURE MODELS AND LANGUAGE PROCESSING

---

- For example, let's say we have three topics: sports, business, and science.
- For each topic, we uncover the most likely words to come from them:

```
sports: [football: 0.3, basketball: 0.2, baseball: 0.2, touchdown: 0.02 ... genetics: 0.0001]
```

```
science: [genetics: 0.2, drug: 0.2, ... baseball: 0.0001]
```

```
business: [stocks: 0.1, ipo: 0.08, ... baseball: 0.0001]
```

- For each word and topic pair, we learn some probability:  
 $P(\text{word}|\text{topic})$ .

---

# MIXTURE MODELS AND LANGUAGE PROCESSING

---

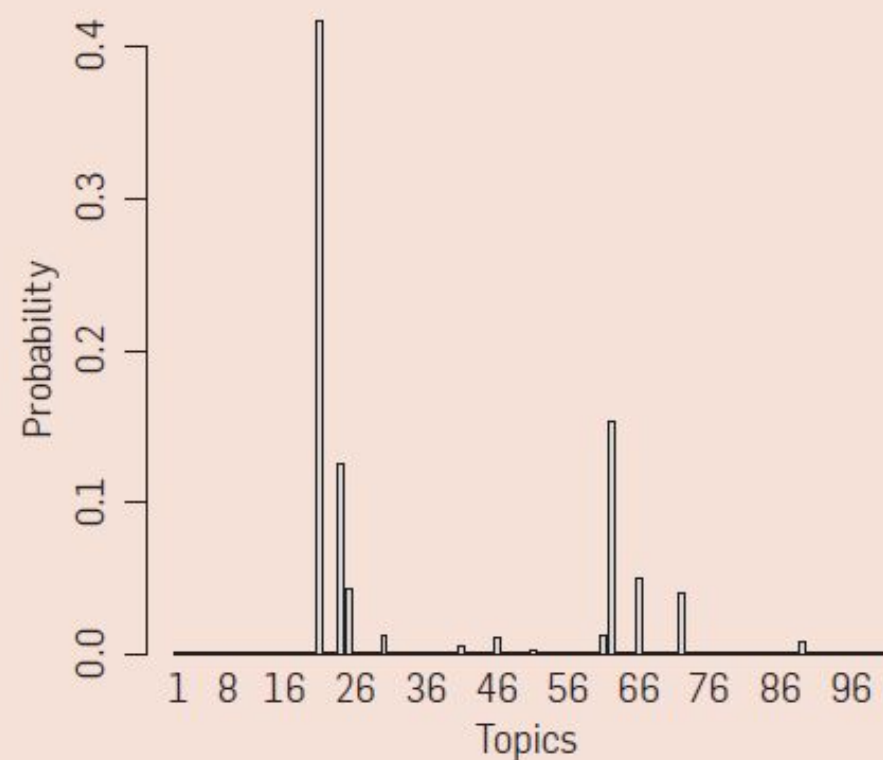
- The *topic distribution* is a multinomial distribution for each document representing what topics are most likely to appear in that document.
- For all our of sample documents, we have a distribution over {sports, science, business}.

ESPN article: [sports: 0.8, business: 0.2, science: 0.0]

Bloomberg article: [business: 0.7, science: 0.2, sports: 0.1]

- For each topic and document pair, we learn some probability,  $P(\text{topic} | \text{document})$ .

# MIXTURE MODELS AND LANGUAGE PROCESSING



“Genetics”	“Evolution”	“Disease”	“Computers”
human	evolution	disease	computer
genome	evolutionary	host	models
dna	species	bacteria	information
genetic	organisms	diseases	data
genes	life	resistance	computers
sequence	origin	bacterial	system
gene	biology	new	network
molecular	groups	strains	systems
sequencing	phylogenetic	control	model
map	living	infectious	parallel
information	diversity	malaria	methods
genetics	group	parasite	networks
mapping	new	parasites	software
project	two	united	new
sequences	common	tuberculosis	simulations

---

# MIXTURE MODELS AND LANGUAGE PROCESSING

---

- 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 to the ‘model’:
  - a. Supervised topic models guide the process with pre-decided topics.
  - b. Position-dependent topic models ignore which words occur in which document and instead focus on *where* they occur.
  - c. Variable number topic models test different numbers of topics to find the best model.

---

## INTRODUCTION

---

# WORD2VEC



---

# WORD2VEC

---

- *Word2Vec* is another unsupervised model for latent variable NLP.
- It was [originally released by Google](#) and further [refined at Stanford](#).
- This model creates *word vectors*, multidimensional representations of words.  
  
assembly  $\rightarrow$  [0.12315, 0.23425, 0.89745324, 0.235234, 0.234234, ...]
- This is similar to having a distribution of concepts or topics that the word may come from.

# WORD2VEC

---

- If we take our usual document-word matrix and take its transpose, 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.

---

# WORD2VEC

---

- ▶ 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' ... and millions more]
```

- ▶ We could make a feature or column for each of these contexts.
- ▶ We could represent 'Paris' in a sparse feature with all possible contexts.

---

# WORD2VEC

---

- Additionally, 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 want to use **dimensionality reduction** to find a *few* concepts per word instead of *all* possible contexts.

---

## WORD2VEC

---

- 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 can replace the ['\_ is a city', '\_ is a capital', 'I flew into \_ today'] columns by a single column, 'IsACity'.

# WORD2VEC

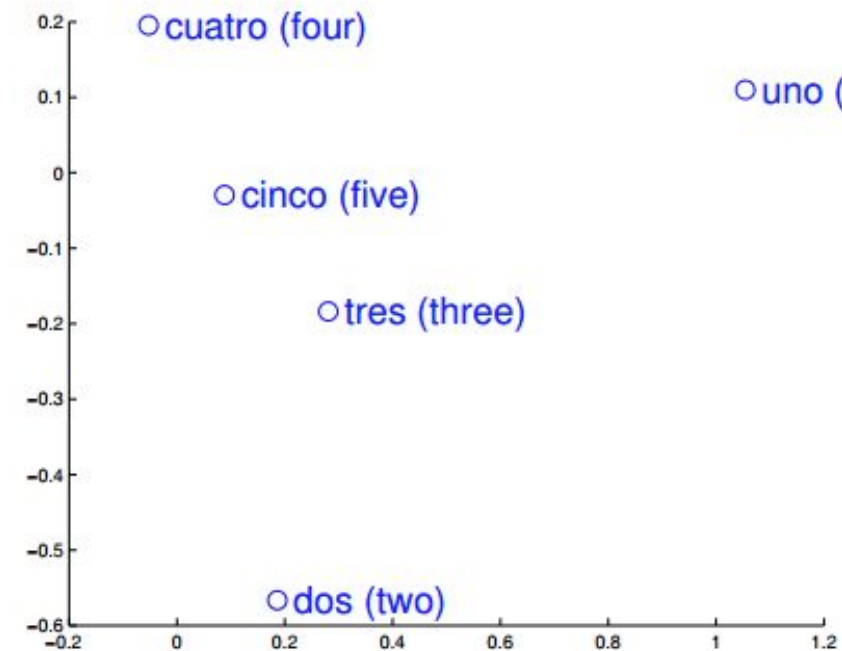
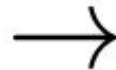
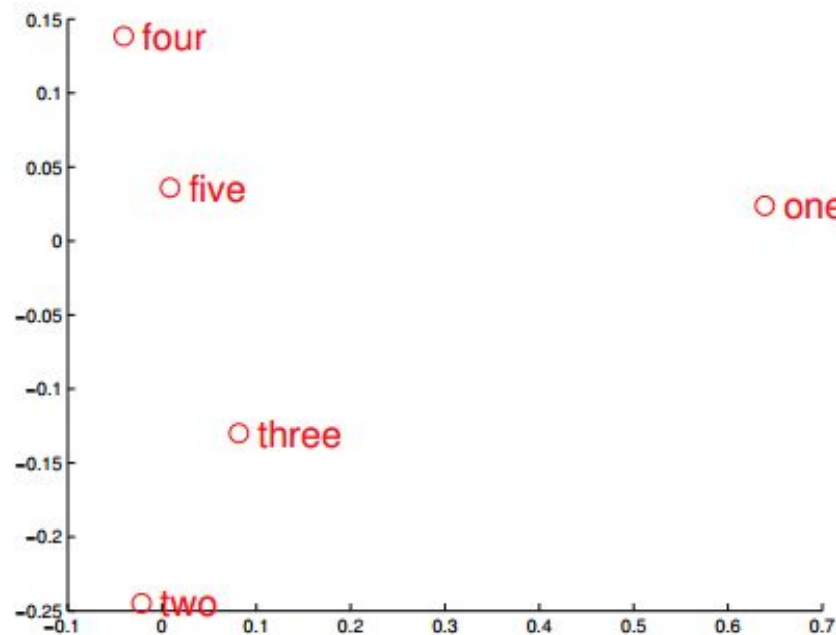
---

- 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

# WORD2VEC

- If we have data for other languages, Word2Vec could also be used for translation.



---

# ACTIVITY: KNOWLEDGE CHECK

---

## ANSWER THE FOLLOWING QUESTIONS



### EXERCISE

1. After reviewing the analogies, brainstorm some word vector math.

For example, what do you think would happen if you subtracted the vector for ‘Man’ from ‘King’. Do you think you would get ‘Queen’?

## DELIVERABLE

Answers to the above questions