Topic Models

topic_models.R

Bayesian Methods

Simple

- Naive Bayes, a "set the baseline" method
- Introduces common independence assumption used in other models

Complex

- Topic models, a hierarchical modeling approach
- Example of a probabilistic generative model
- Unsupervised, like LSA
  - Supervised version also available
- Linked to vector space models

Naive Bayes

Classification problem

Assign class label to Y given collection of categorical indicators (e.g., word present/absent)

Assign to category $\hat{Y}$ that maximizes conditional probability

$$\max_y P(Y=y \mid X_1, X_2, \ldots X_k)$$

Complication

Suppose k is very large, possibly larger than number of obs

- Lack enough examples to build conditional probability from frequencies

Example: Federalist papers

- 75 documents, but 10,000 word vocabulary

Naive Bayes is competitive in cases with few training examples

- Provided its assumptions hold

Naive Bayes Solution

Employ Bayes rule

$P(Y|X)\,P(X) = P(X|Y)P(Y) \;->\; P(Y|X) = P(X|Y)P(Y)/P(X)$

$\max_y P(Y=y|\,X_1, X_2, \ldots X_k) = \max_y P(X_1, X_2, \ldots X_k|Y)\,P(Y)$

Assumptions

Know prior probabilities (such as equal!)

$\max_y P(Y=y|\,X_1, X_2, \ldots X_k) = \max_y P(X_1, X_2, \ldots X_k|Y)$

X_j are conditionally independent given Y

$\max_y P(Y=y|X_1, X_2, \ldots X_k) = \max_y P(X_1|Y)\,P(X_2|Y)\cdots P(X_k|Y)$

Rationale in language

Reduces problem to product of frequencies from 2x2 contingency tables in case of words/text

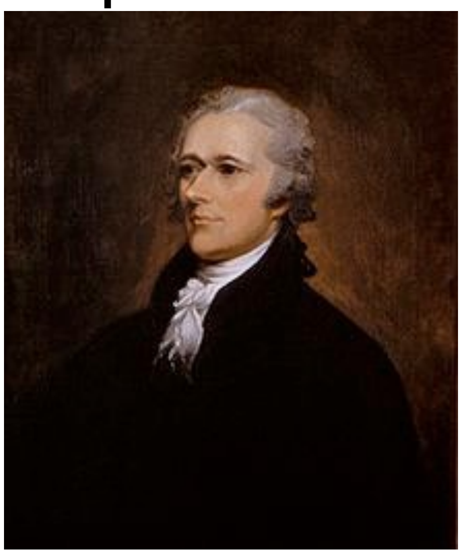
Example: Federalist Papers

Federalist papers

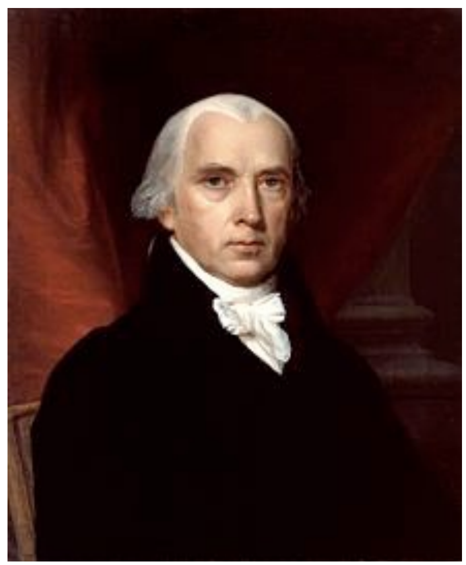
85 essays advocating US Constitution in 1787-1788

Revisit text by Mosteller and Wallace (1964)
Who wrote the 12 disputed Federalist papers?

Supervised classification



Hamilton
51



Madison
14

Jay
5

Hamilton & Madison
3

Wharton
Department of Statistics

Federalist Papers

Data

Nothing fancy: a CSV file

Elaborate data processing needed for web-scale applications

Three "variables" for each of 85 documents

author, number, text

Sample

To the People of the State of New York: AFTER an unequivocal experience of the inefficacy of the subsisting federal government, you are called upon to deliberate on a new Constitution for the United States of America…

Preprocessing

Downcase

Want a document-term matrix for identifying useful words

Results of Naive Bayes

Simple analysis

Identify whether a word appears or not (0/1) rather than count

Component probabilities $P(X_W|Y)$ reduce to relative frequency of a word appearing in the papers written by each author

Which words to use

Words that are reasonably common

Avoid words that appear in every document.

Avoid words that don't get used by an author.

What about the prior probability?

Compare to other classifiers

Topic Models

Conceptual model for the generation of text

Text expresses an idea or “topic”

Presidential address might move from domestic economics to foreign policy to health care.

Current topic determines the chances for various word choices

The words “inflation” or “interest rate” are more likely to appear when discussing economic policies rather than foreign policy

Hierarchical model

Identify the number of topics

Define a probability distribution for each

Each document mixes words drawn from topics

Conditional independence, given topic (naive Bayes)

Heuristic Motivation

Each document mixes words from collection of topics

topic = probability distribution over words

Original details: Blei, Ng, and Jordan 2003

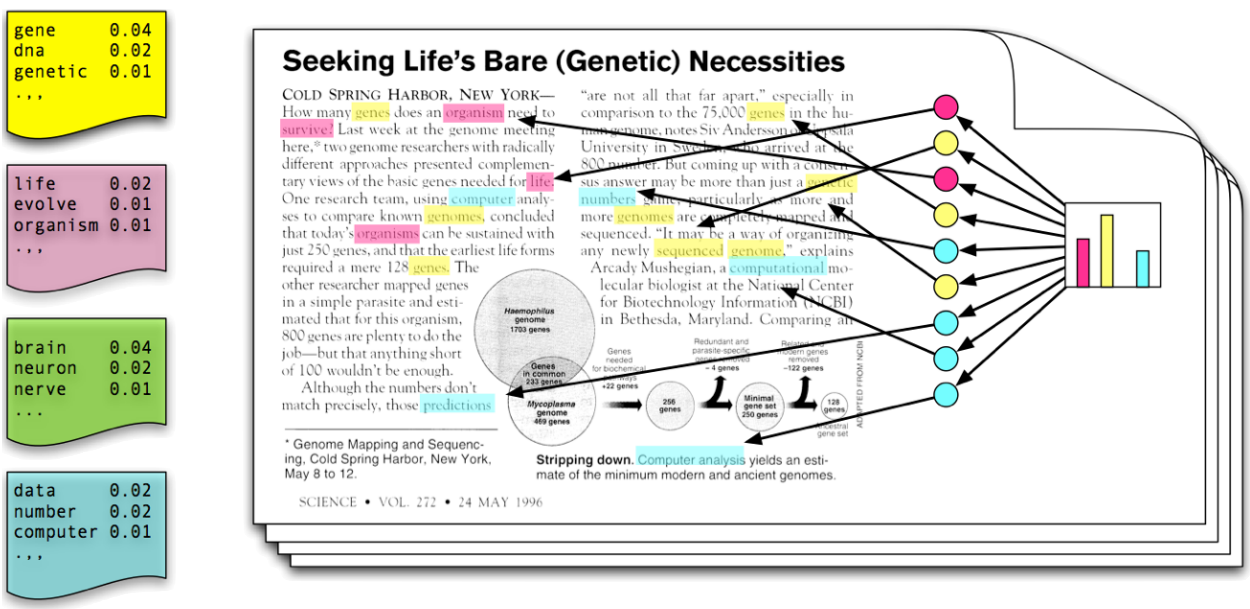


Figure from Blei, Intro to Topic Models

Wharton
Department of Statistics

Probability Model

Beta:Binomial
as
Dirichlet:Multinomial

Latent Dirichlet allocation (LDA)

Define K topics

Discrete dist over vocabulary $P_k \sim \text{Dirichlet}(\alpha), k = 1,\ldots, K$

Parameter α controls sparsity of the distribution

Each document mixes topics

Distribution over topics in doc_i $\theta_i \sim \text{Dirichlet}, i = 1,\ldots, n$

θ_i are probabilities

Word probability $P(w \text{ in doc } i) = P_k(w)$ $k \sim \text{Multi}(\theta_i)$

Number of words within doc allowed to be random/fixed

Wharton
Department of Statistics

Expected Word Counts

Matrix product determines counts

Let K x m matrix P denote the matrix with probability distribution P_k in the k^{th} row.

Let the nxK matrix T denote the mix of topics in the documents, with the mix for document i in row i.

Then the expected number of word tokens of type j in document i is $(T\ P)_{ij}$.

Factorization

Topics models imply a factorization of the expected count matrix, the document term matrix C

$$E(C) = n_i\ T\ P$$

and the SVD is one way of factoring C!

Example

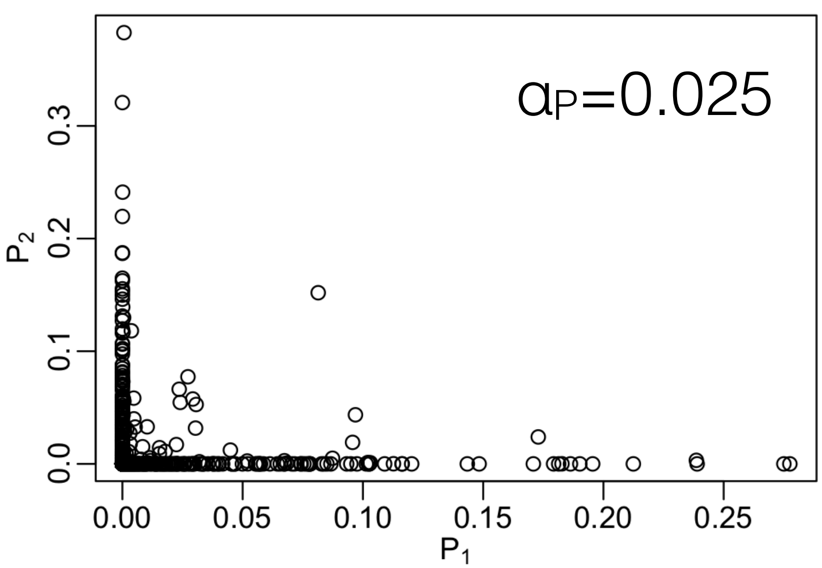
Simulate data from a topic model

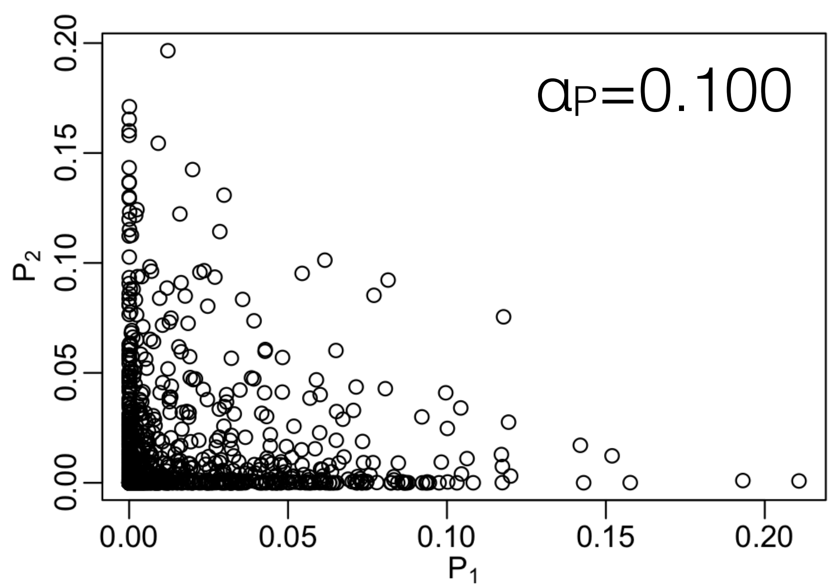
Pick the number K of topics

Pick size m of the vocabulary and the number of documents n

Choose α_P that controls “sparsity” of topic distributions

Small α_P produces nearly singular distributions with little overlap.





α_p = 0.025 in following

Simulate the Documents

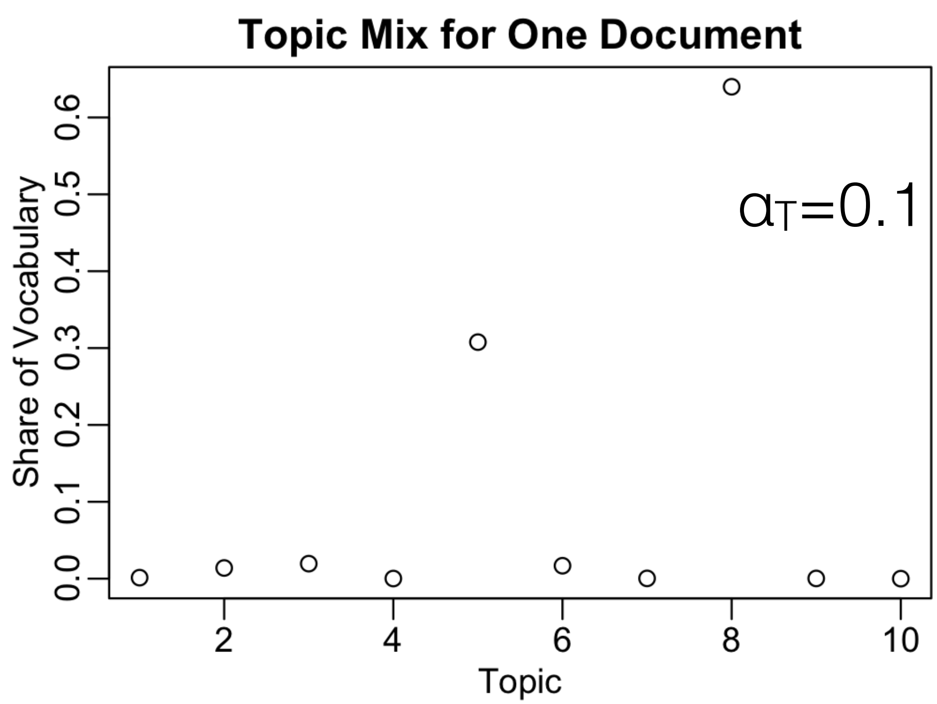
n=5000
m=1000
$n_i \approx 100$

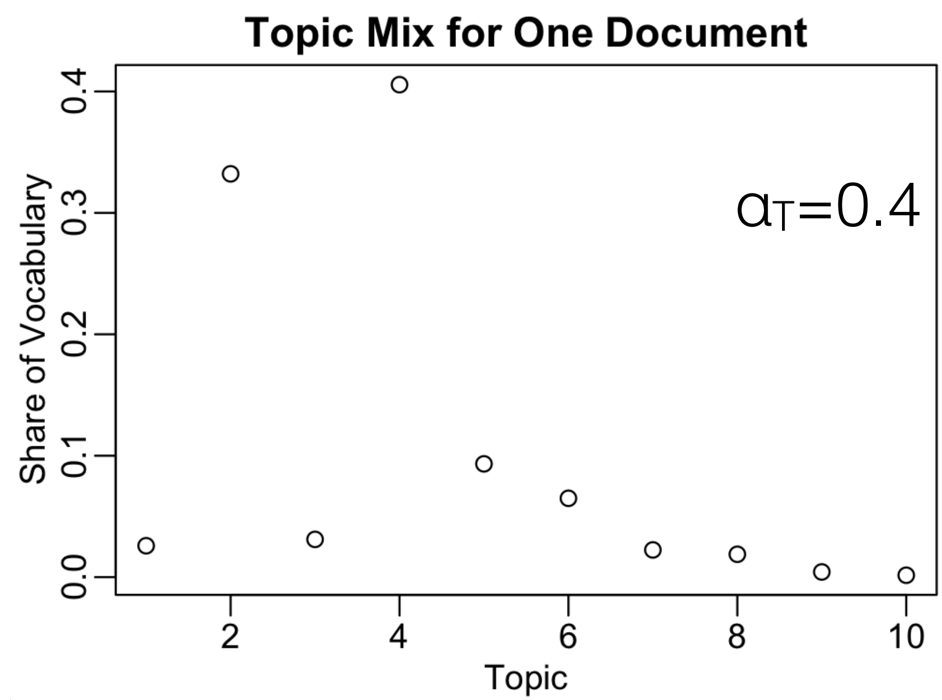
Generate documents

Choose average length of documents (poisson distribution)

Pick α_T that controls the mix of topics within documents

Small α_T produces documents predominantly of one topic.



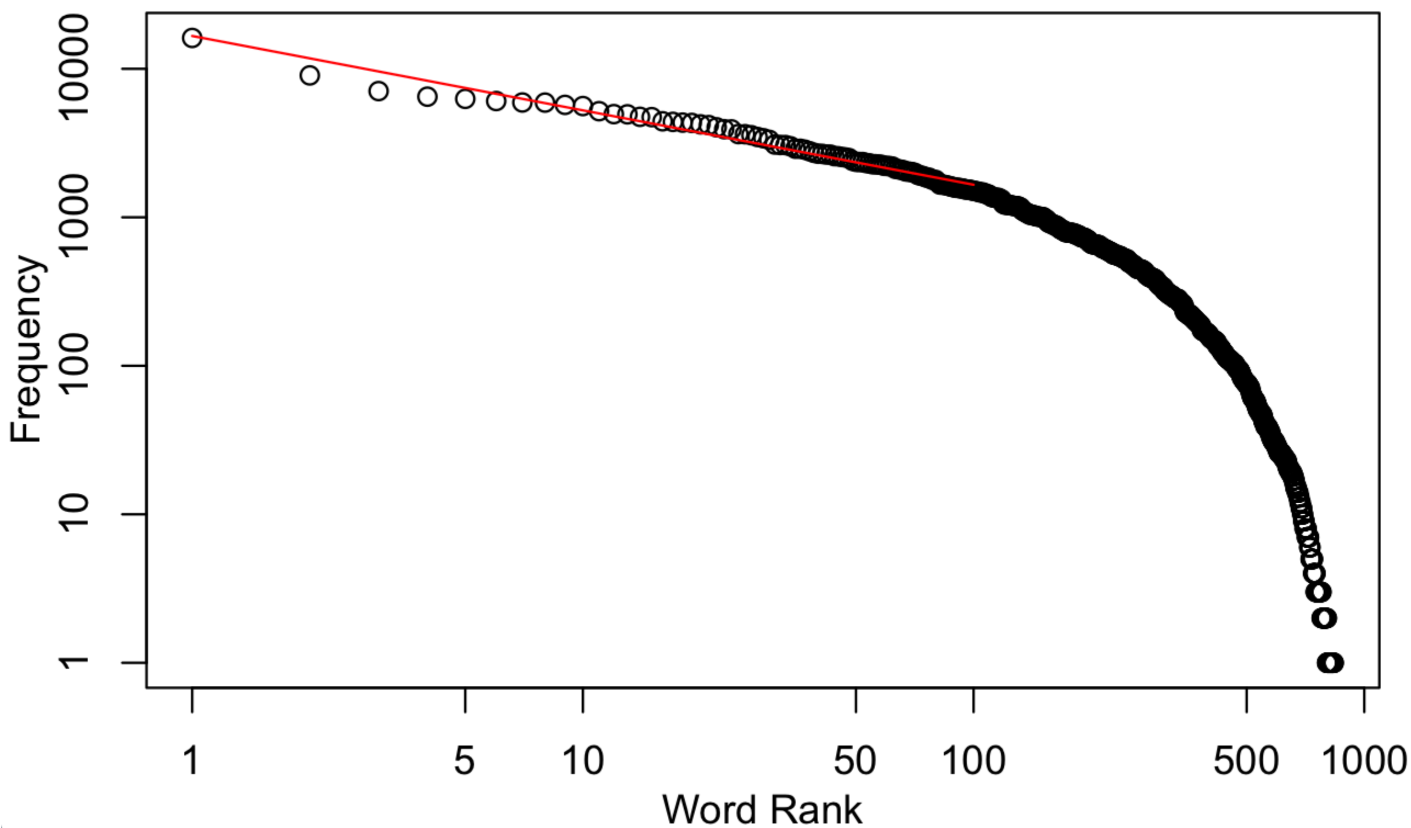


α_T = 0.4 in following

K=10 topics

Word Frequencies

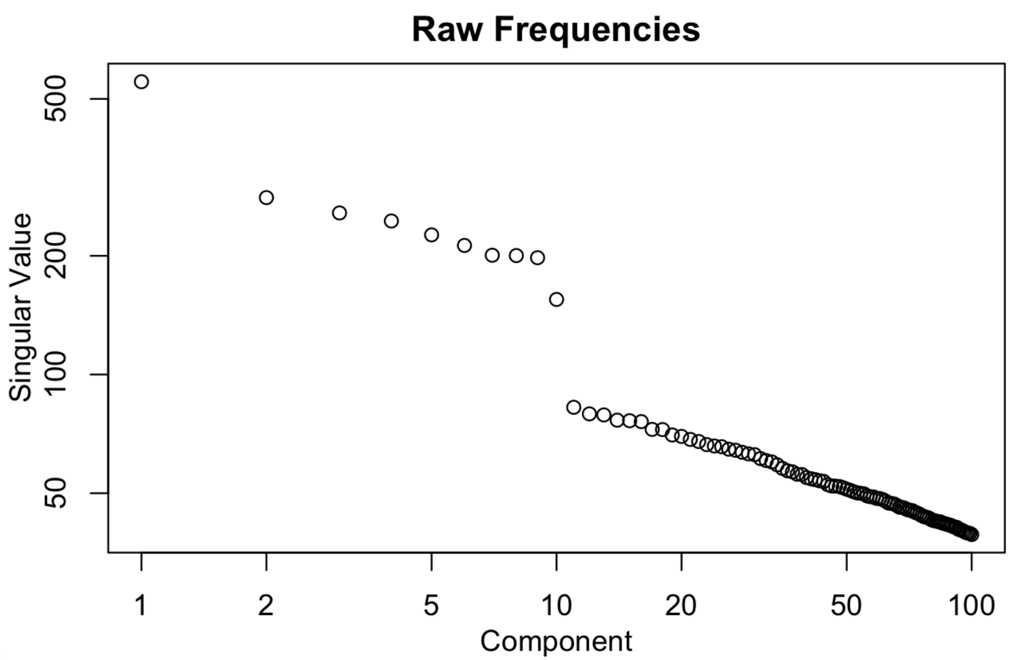
Typically not very close to Zipf as we find in real text

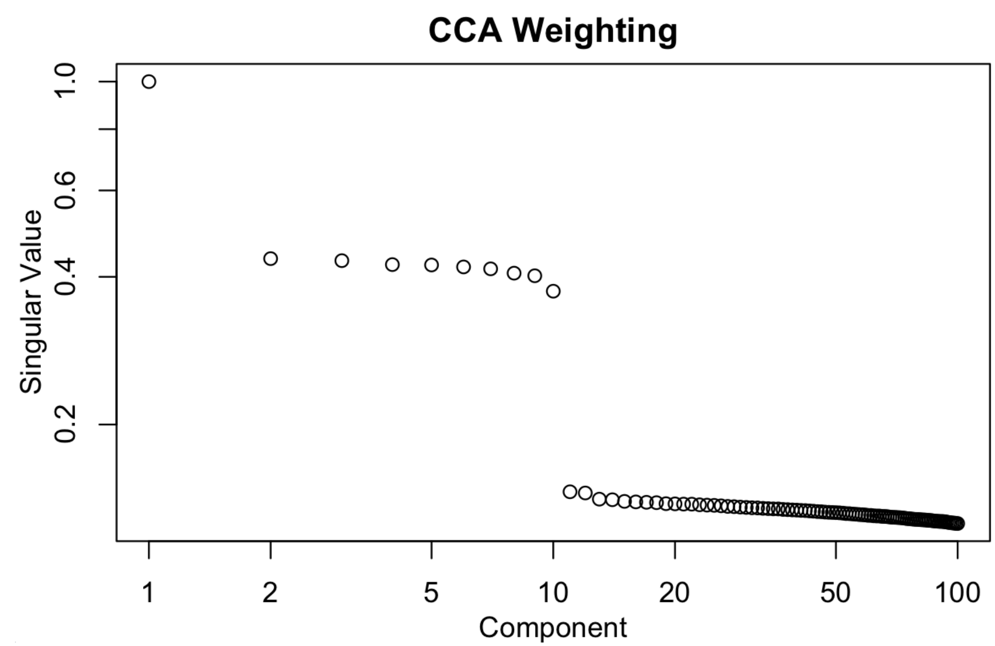


LSA Analysis

Compute the SVD of the counts

- Raw counts and using CCA weights
- Number of topics stands out clearly, particularly in CCA



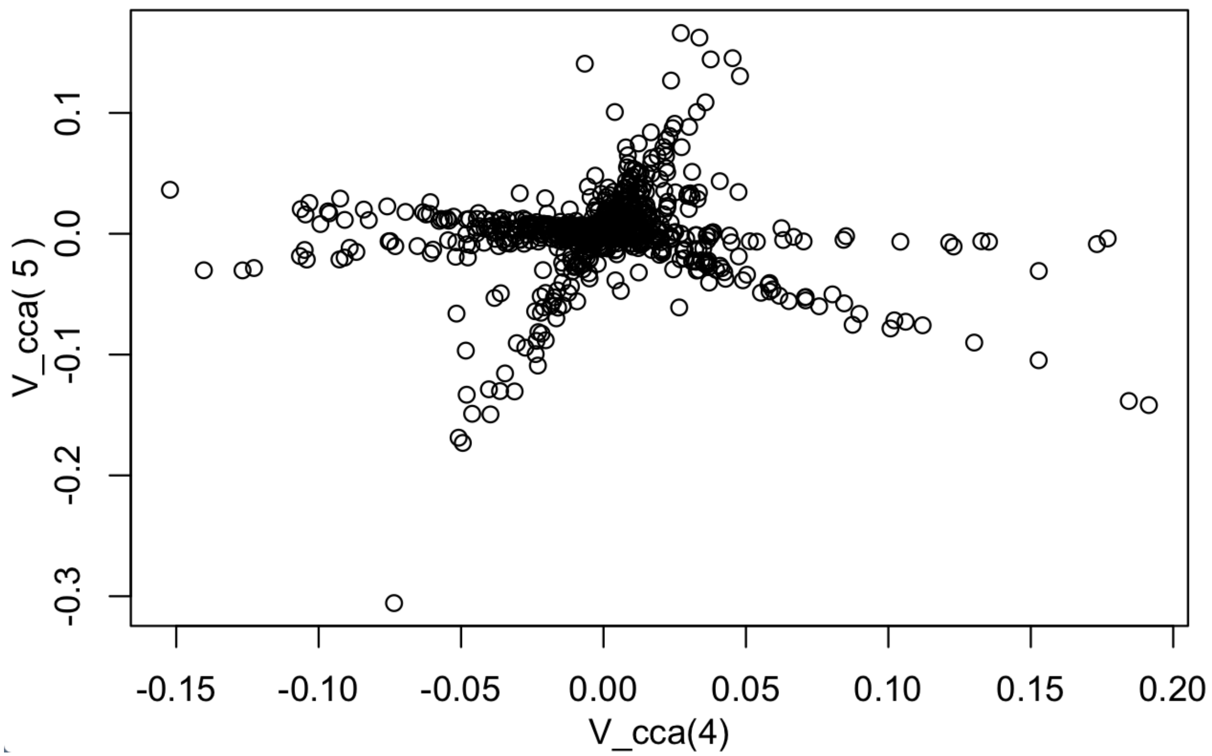


LSA Analysis

Loadings have the “ray-like” behavior

Similar to those in LSA analysis of wine tasting notes

More clearly defined



Topic Model Analysis

Same simulated data

Pick number of topics (e.g., know there are 10)

Input the associated DTM

Results

Indicates which topics most prevalent in documents

Associates word types with the discovered topics

Goodness-of-fit

Obtain overall log-likelihood of fitted model

Vary the number of topics to see how fit changes

Topic Models: Wine

Fit topic models to the data set of wine tasting notes

Use all 20508 documents, with 2659 word types

after removing/merging the OOV types

Fit with K=10 topics

Topics in documents

Lists topics comprising the tasting notes

```
> top[,1:15]
     1  2  3  4 5  6 7  8  9 10 11 12 13 14 15
[1,] 1 10  5  5 8  5 5  5  9  5  5  7  5  9  9
[2,] 3  5 10  9 9  9 7  9 10  9  2  4 10  5 10
[3,] 6  8  2 10 5 10 4 10  8  2  4  8  4 10  8
[4,] 9  9  3  4 3  8 6  7  1  3  1  6  9  4  5
[5,] 7  2  7  2 4  2 8  2  5  1 10 10  8  8  4
```

Word types in topics

Not real exciting…

Documents too short?

```
      Topic 1   Topic 2   Topic 3   Topic 4   Topic 5
 [1,] "and"     "comma_"  "and"     "comma_"  "comma_"
 [2,] "dash_"   "period_" "dash_"   "period_" "dash_"
 [3,] "medium"  "and"     "medium"  "dash_"   "finish"
 [4,] "period_" "dash_"   "period_" "and"     "aromas"
 [5,] "with"    "medium"  "aromas"  "with"    "bodied"
 [6,] "comma_"  "entry"   "fruit"   "entry"   "full"
 [7,] "body"    "with"    "leads"   "aromas"  "period_"
 [8,] "bodied"  "full"    "body"    "body"    "fruit"
 [9,] "aromas"  "body"    "fruity"  "follow"  "medium"
[10,] "acidity" "fruit"   "this"    "good"    "entry"
```

Unsupervised Modeling

Pretend we don't have a response.

Do frequencies of words reveal clusters?

Unsupervised model

- No response variable
- Which documents are similar

Document similarity

- Data is very sparse:
  - 2659 types (OOV) but only ≈37 tokens in doc

Random projection preserves distances

Word Embedding
n-grams

Bigrams, n-Grams

Document term matrix

Associates words that appear in same "context"

A document defines the context

Natural association for modeling a property of a document

n-Gram matrix

Bigram: The adjacent word defines the context

Trigram: The adjacent words to either side define the context

n-gram: Use varying numbers of adjacent words

Designed to study the relationship of words

Return to Token Space

Bigram matrix origins

Consider two matrices with elements 0 and 1

Total number of rows in each = total number of word tokens -1

prior word type

	w_1	w_2	w_3	…	w_m
t_1	0	0	0		0
t_2	0	0	1		0
t_3	1	0	0		0
t_4	0	1	0		0
t_5	0	1	0		0
	0	1	0		1
⋮					
t_{N-1}	0	0	0		0
t_N	0	0	0		1

W_{-1}

word types

	w_1	w_2	w_3	…	w_m
t_1	0	0	1		0
t_2	1	0	0		0
t_3	0	1	0		0
t_4	0	1	0		0
t_5	0	0	0		1
⋮					
t_{N-1}	0	0	0		1
t_N	0	0	0		0

W

N = total # tokens
m = # word types

Bigram Matrix

Matrix calculation

Matrix product times its "lag"

$$B = W_{-1}^T W$$

so that

$$B_{ij} = \#\{\text{token of word type } w_i \text{ precedes } w_j\}$$

B is an m x m matrix, where m = size of vocabulary

Interpretation as covariance

Consider the rows of the Nxm matrix W as flowing over time

stochastic process that picks the words

$B_{ij} = N \operatorname{cov}(w_i, w_j)$

again, ignoring the mean values that will be very close to 0

Word order matters!

Bigram Matrix

Standardization

- Word types that are more common will tend to co-occur more often than word types that are more rare
- Weighting, such as CCA or td-idf, are common
  - CCA divides by square root of the product of the type frequencies
- CCA weights convert the covariance into a correlation
  - approximately, because sqrt(m_j) ≈ sd(j^{th} column of W)

Tokenization

- Key choices remain highly relevant
- Stemming, removing punctuation, handling OOVs

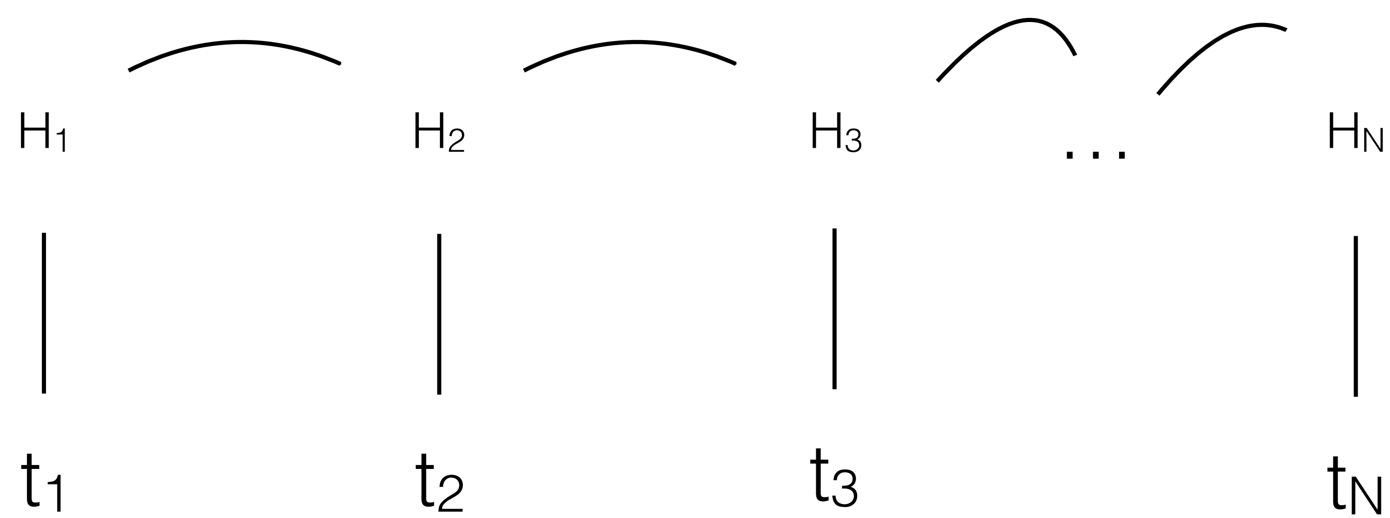
Bigrams and Models

Hidden Markov model

Imagine underlying language communicates sequence of ideas or concepts, say H_k, for k = 1,…, K

Each concept is associated with a certain vocabulary of words, say Vk.

We can learn about the concepts by discovering words that tend to occur near each other, or be used in the same way.



Word Embedding

Theory

SVD of the bigram matrix B reveals aspects of hidden states

Conversion using "thin" SVD

Retain some of the components of the SVD of bigram matrix (after standardizing)

$$\underset{m\times m}{B} \text{ –> } UDV^T$$

Suppose we retain d components, then the rows of U (an m x d matrix) provide an embedding of words in a d-dimensional, real-valued space.

Random projection ideas are typically necessary for handling a large corpus with a diverse vocabulary ($m \approx 100{,}000$ or more)

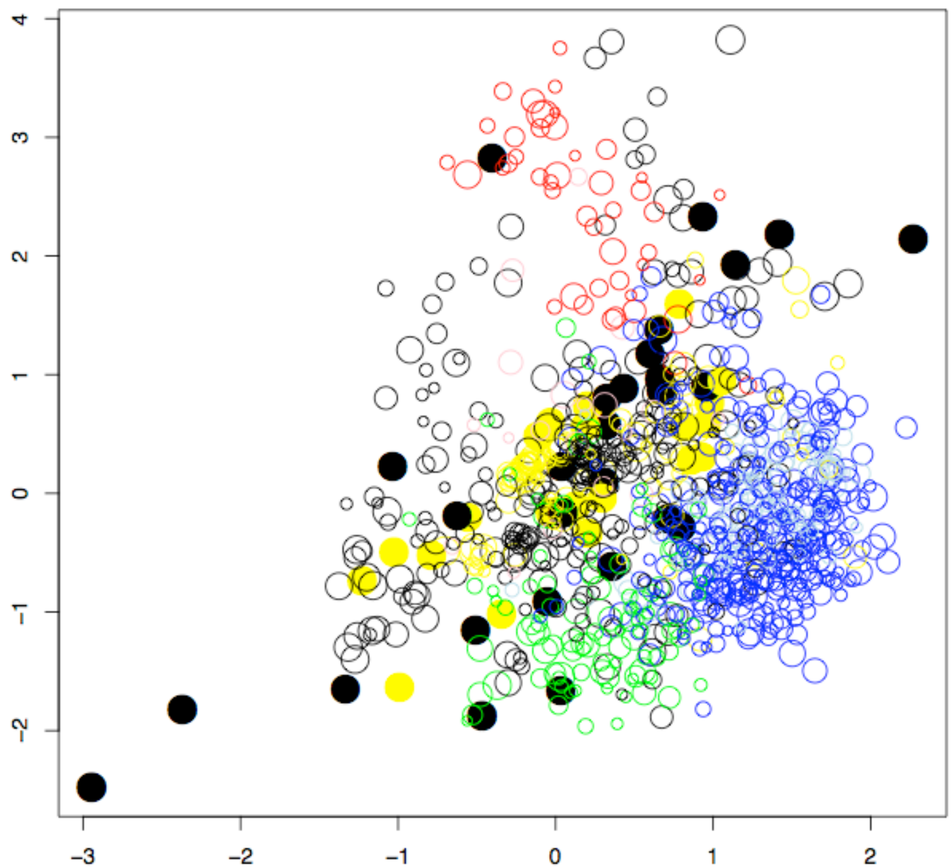
Examples of Embeddings

Parts of speech

Obtained from analysis of much larger corpus

Regular text rather than domain specific text like wine reviews

noun
verb
adj
unk

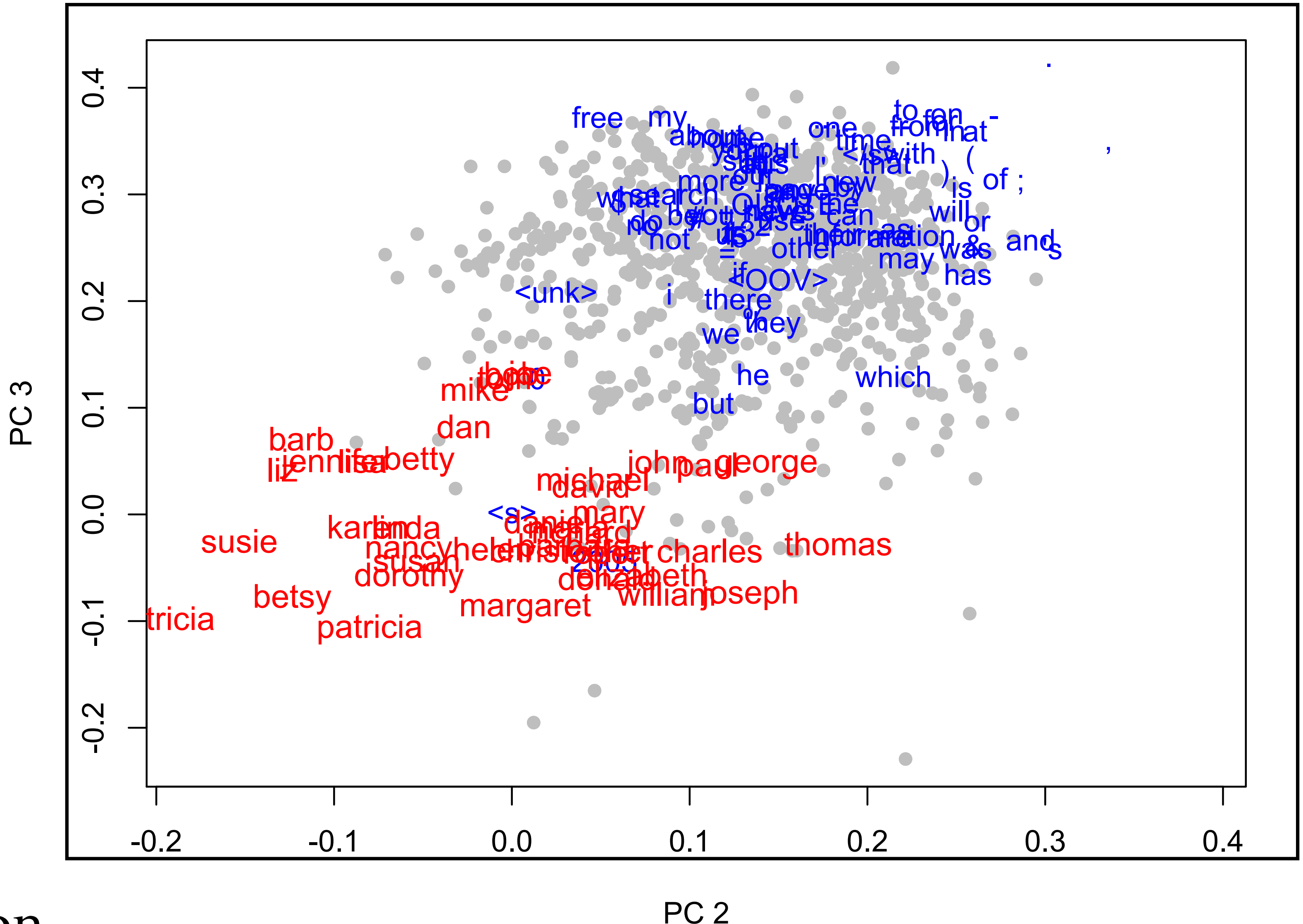


OOV in black

Wharton
Department of Statistics

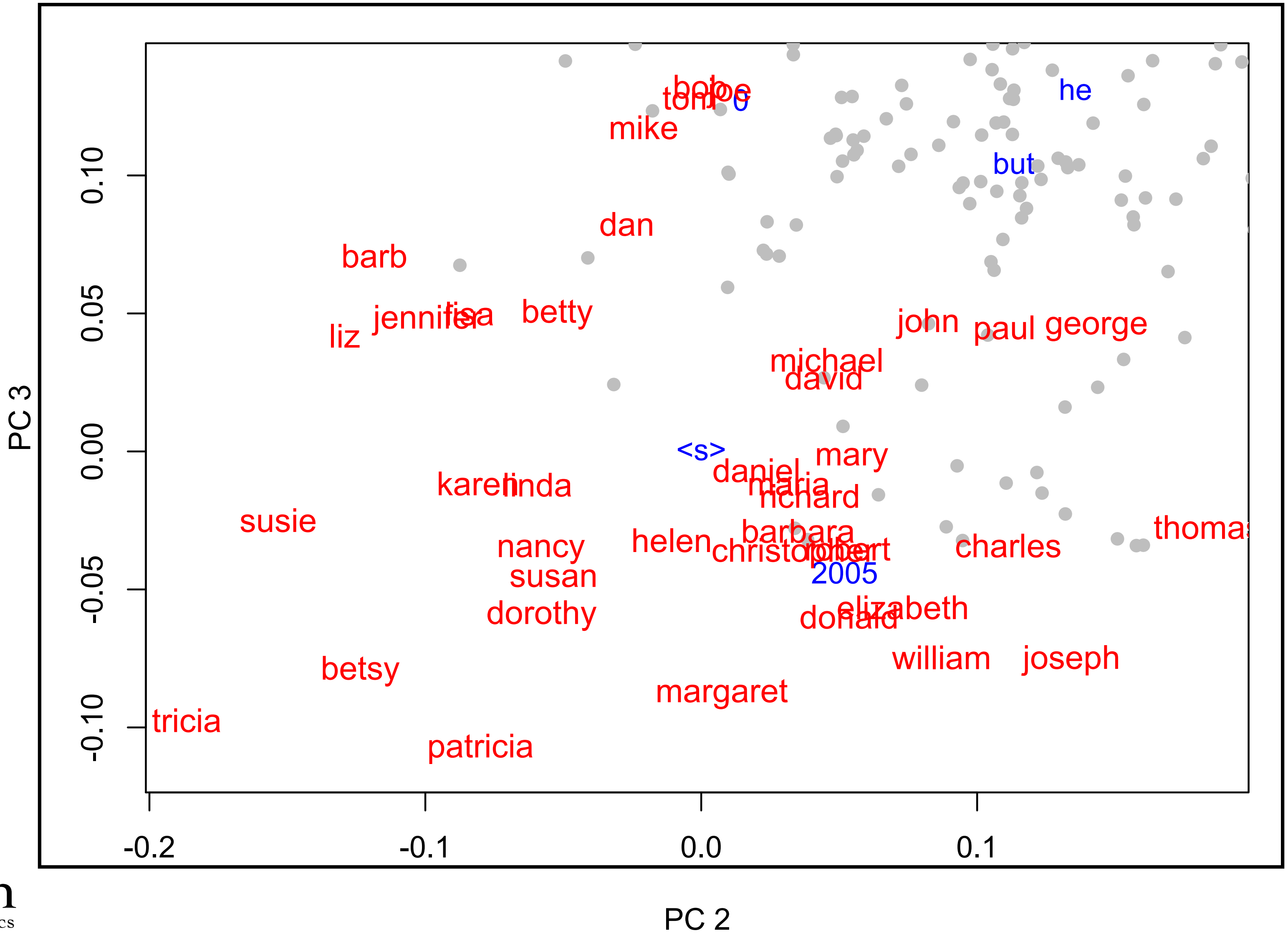
Examples of Embeddings

Plot of two singular vectors



Examples of Embeddings

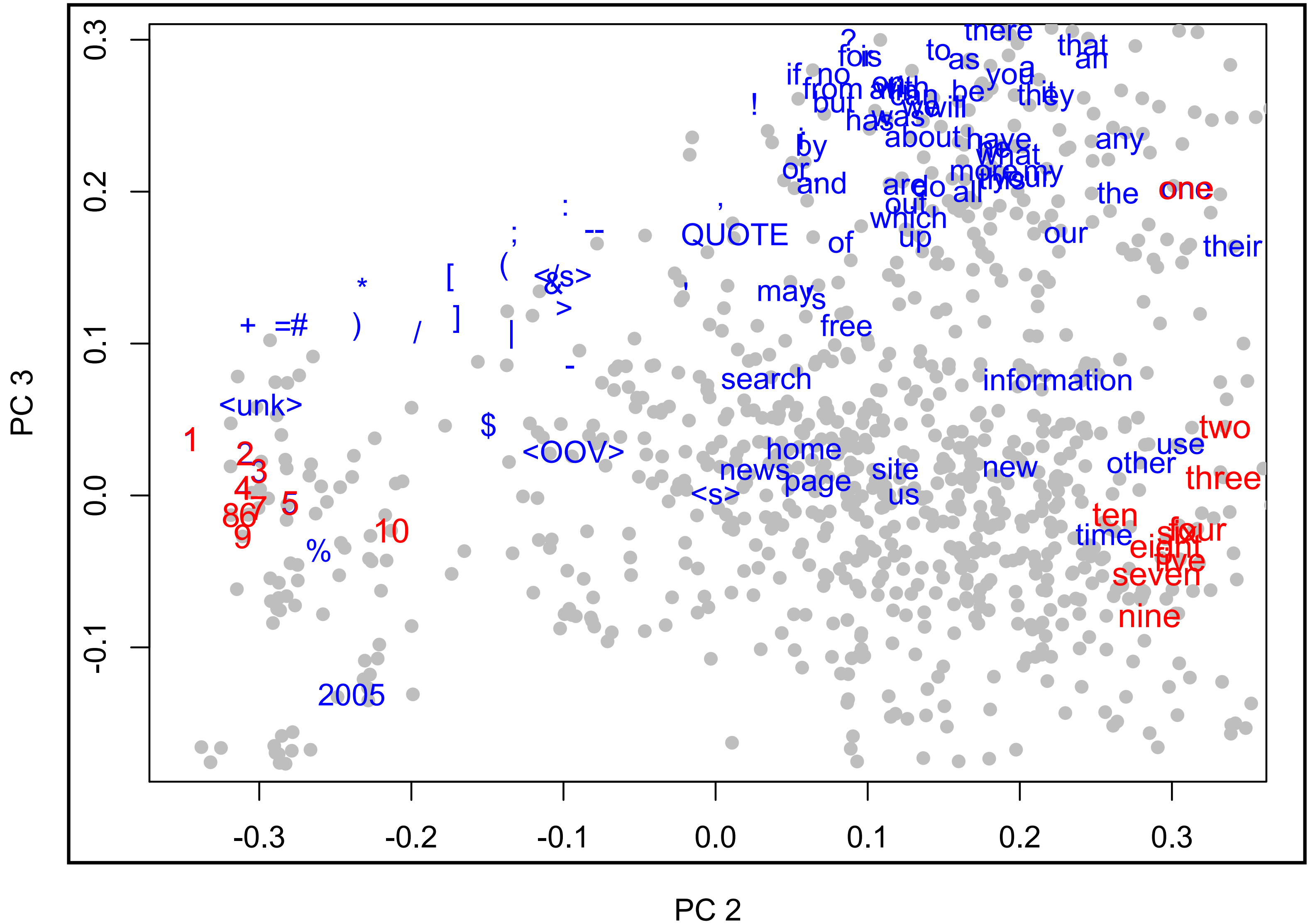
Zoomed in view of same singular vectors



Wharton
Department of Statistics

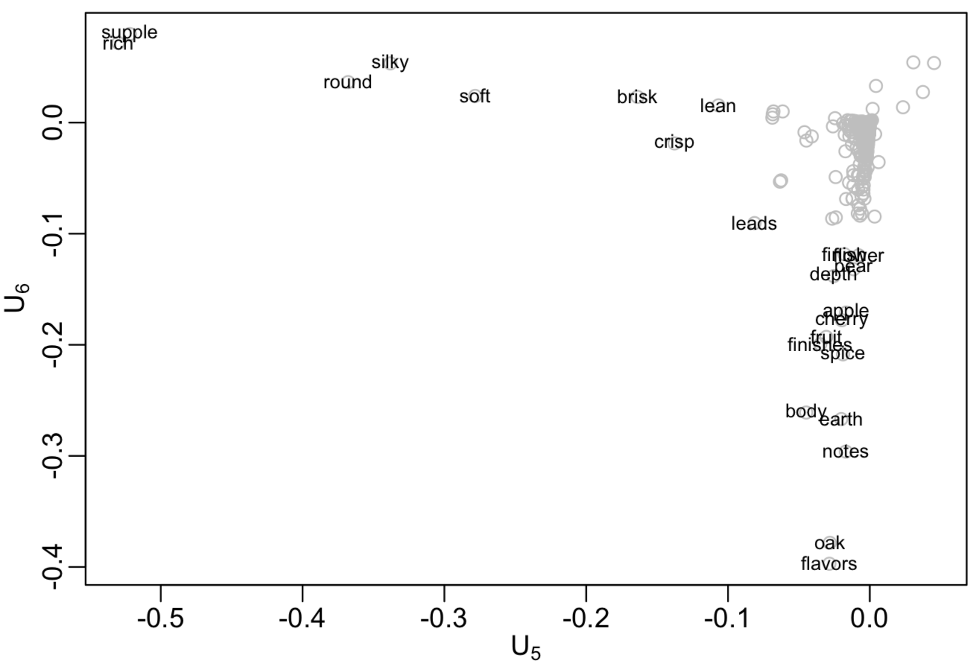
Examples of Embedding

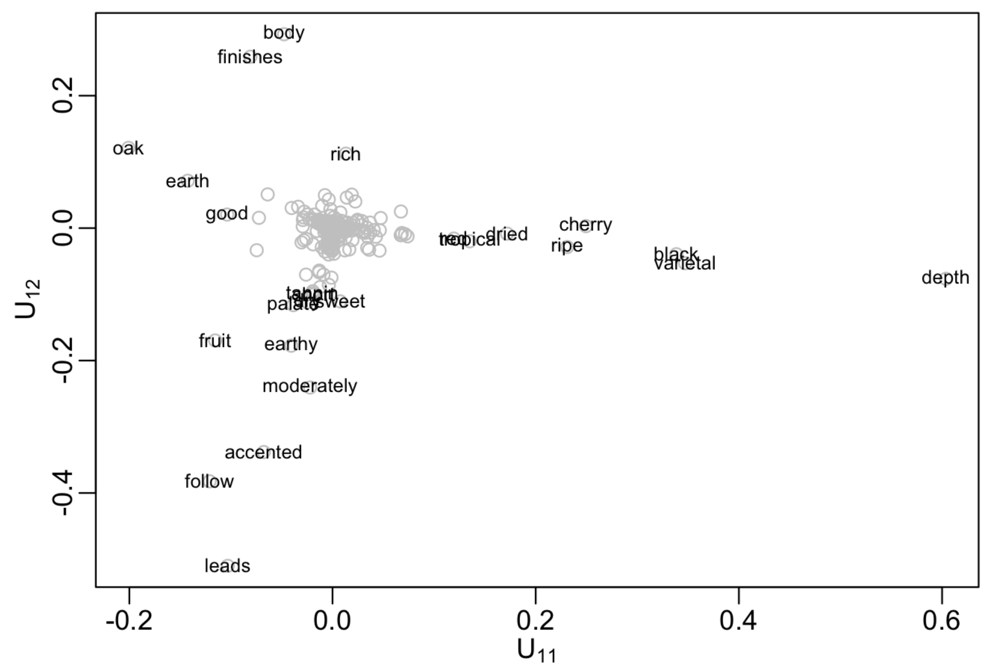
Numbers as words and digits



Bigrams in R

Typically weighted, but worked better here with small corpus to leave raw counts.





Word2vec

Alternative approach to word embedding

Derived from “deep” neural network

Motivating probability model

Build a model for $P(W_t|W_{t-1}, W_{t-2}, \ldots)$

Output a probability distribution over next word

Bigram case has one preceding word for the context

Popularity

Algorithm for solving large neural network

Fast implementation, very effective demonstration

Word2vec Structure

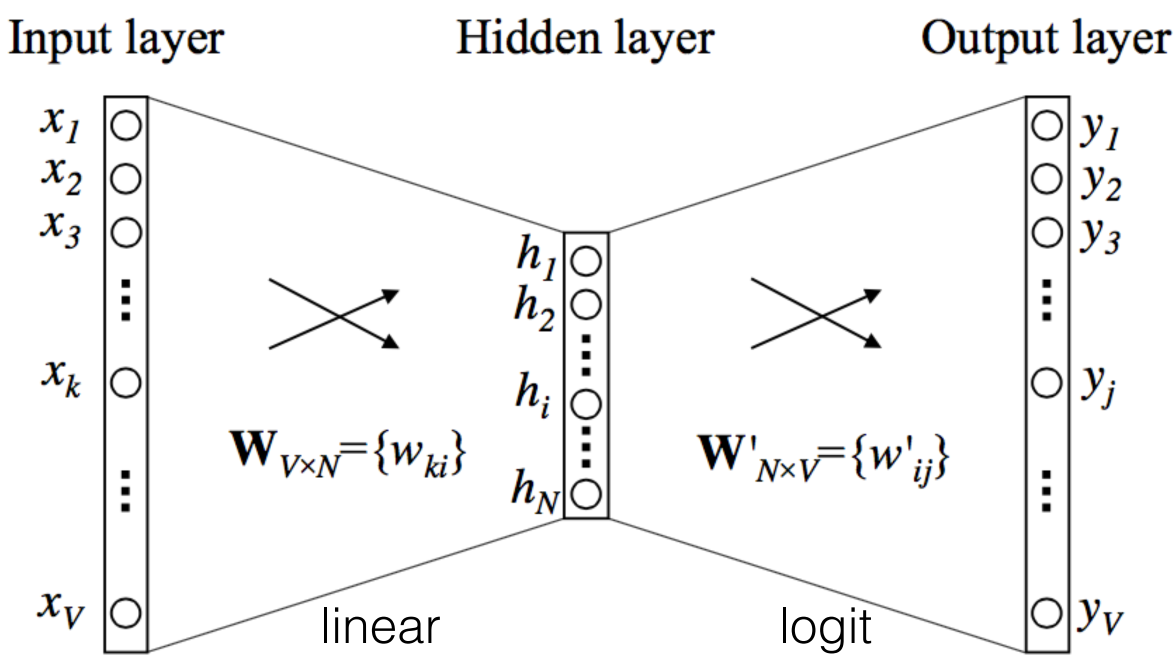
Deep network

Network structure

Input x is dummy word indicator, $(x^TW) = h^T$ hidden state

Output "softmax" $y_j = \exp u_j / \text{sum} \exp u_j$, $u_j = (h^TW')_j$

one-hot encoding



probability distribution

large number of parameters to estimate

Rong, "word2vec parameter encoding explained"

Idea of Embedding

Text

The quick brown fox **jumped** over the fence.

context target

Choose vector of coordinates V_C to represent context word and to represent target word V_T so that score $V_C^T V_T$ is high.

Wrong text: The quick brown fox **ate** over the fence.

Choose vector of coordinates V_C to represent context word and to represent WRONG target word V_W so that score $V_C^T V_W$ is small.

Example

Code widely available on Internet

Train during class

- Compute intensive, so I will run on a server back at Penn
- Build N = 200 dimensional hidden state vector
- Loads a corpus to build
- Trains in about 5 minutes

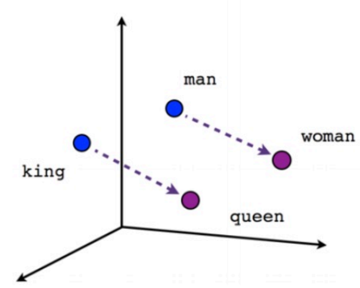
Word analogies

- paris:france :: london: ???
- king:man :: queen: ???

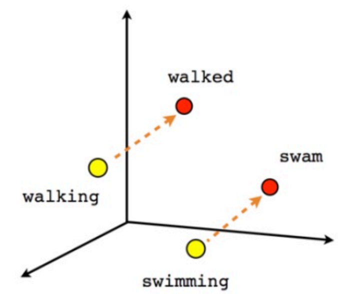
Lots better with much larger corpus

More Examples

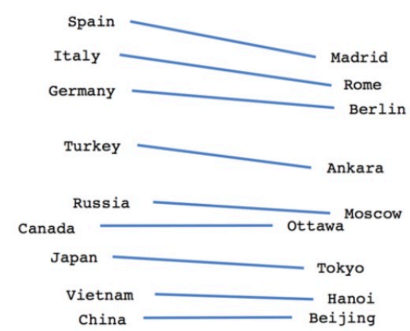
See papers of Mikolov et al (Google)



Male-Female



Verb tense



Country-Capital

from TensorFlow site

Czech + currency	Vietnam + capital	German + airlines	Russian + river	French + actress
koruna	Hanoi	airline Lufthansa	Moscow	Juliette Binoche
Check crown	Ho Chi Minh City	carrier Lufthansa	Volga River	Vanessa Paradis
Polish zolty	Viet Nam	flag carrier Lufthansa	upriver	Charlotte Gainsbourg
CTK	Vietnamese	Lufthansa	Russia	Cecile De

Table 5: Vector compositionality using element-wise addition. Four closest tokens to the sum of two vectors are shown, using the best Skip-gram model.

Deep Learning

Deep Learning

Continuing development of large neural networks in models of language

Recursive neural network

- Sequence to sequence learning
- Used for grammatical error correction, language translation
- Long-short term memory (LSTM) network nodes

Very large networks require substantial computing resources to run in reasonable time

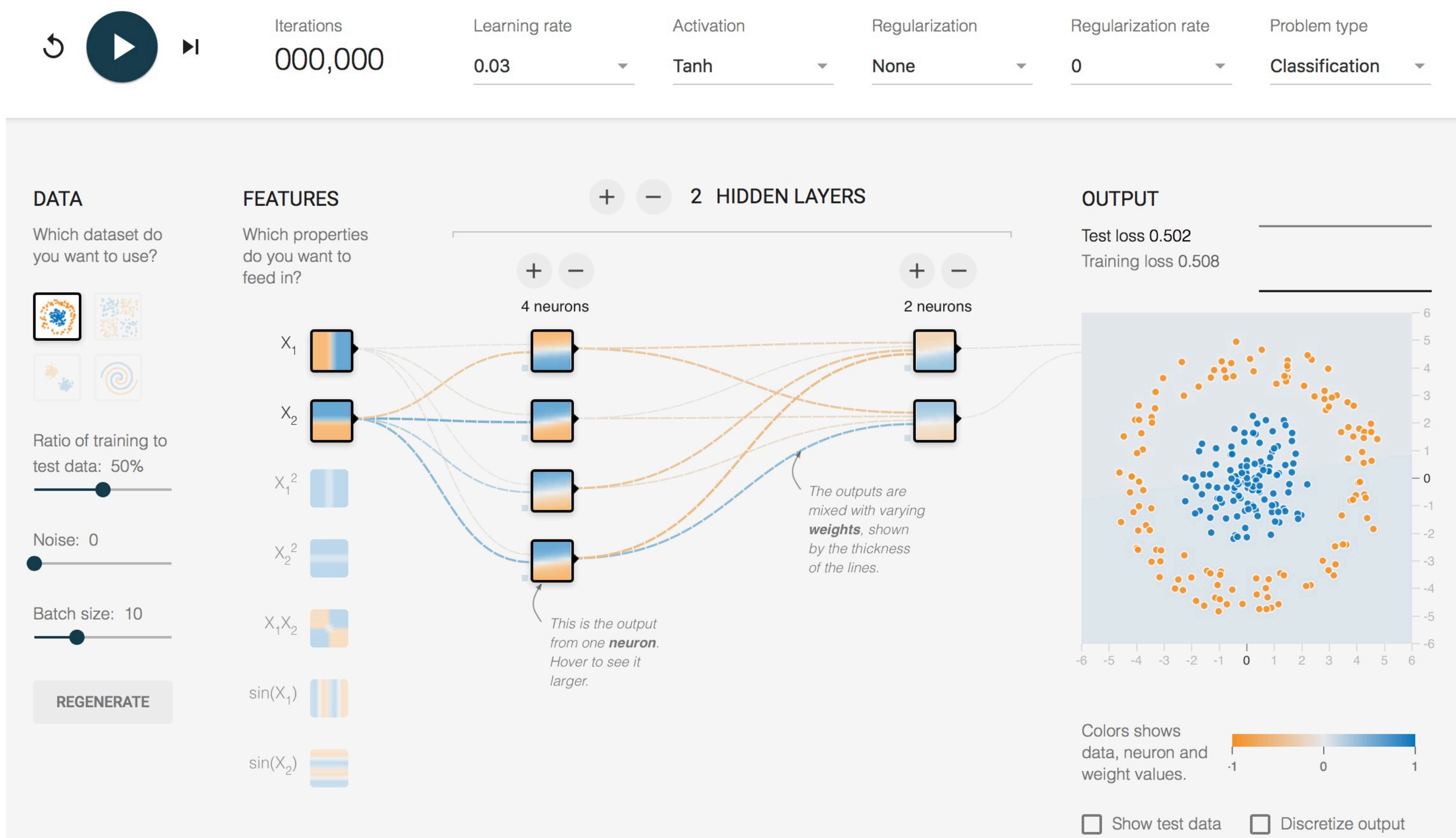
- Commonly built using graphics processors for faster matrix calculations

TensorFlow Animation

Online example of large neural networks

Not for text

Useful to explore flexibility



www.tensorflow.org

Text Examples

Language generator

- Show it lots of examples of language
- Builds a probability model for the language
- Can use to classify language source

Example

- Version of the code from Zaremba “Learning to Execute”
- Build model (takes a while to train on a laptop!)
- Character level generator (not words, it works at char level)
- Need a lot more text for training than the “few” wine reviews

Example: Generating Text

Generate new reviews

remember: character level generator

Can you tell the type of wine being described?

```
Pale golden straw color. Lemon oil, lemon zest, and cinnenfruit flavors. Finishes with a sl
ightly spring, and fruit dry finish. A very nice depth of fruit, this has mouch tarmen and
floral acidity in the finish.
Brilliant yellow hue. Yeast, dried citrus and dried apple and merballoon aromas. Medium-bod
ied, this has aple and crisp tropical fruit and a touch of spice and an-partint valb finish
. A somp, short finish with eleggantly langer.
Pale golden silver color. Rubber bash aromas follow through on a broem, melon fone food win
e.
Golden color. Floral toasty, lemon seaut, and roable  cake rind intession and apple skin Ri
nese.
```

```
Crunchy leafy cherry, blackberry, nutmeg and dried hell fade. A very nice effort
 has distinctive noce-depty medium-full body and a long, zesty, and quaffer.
Creamy berry, mydill-bodied palate with soft tannins, nicely integrated, tangy f
inish with lively thavines, tasty Shyrom marinagek that tasty!
Black cherry, plum, saged oak and singer accented finish. Ample oak and pleasn's
 lacked, well maunteriled rubding's steak.
Limerbinaro, tomato, cherry and black fruit kis aromas follow through on a mediu
m-bodied palate with chewy tannins and lively acidity. A ripe, remonion foods.
```

Example: Scoring Text

Scoring existing text

DNN builds a probability model, so it can assign a likelihood to a review as being a review of red or of white wine.

Feed notes on tasting red wines into both models

```
macpro:~/lua/projects/char-rnn > make test_wine_white
head -n 10 data/wine_red/wine_red_test.txt | th score.lua wine_white/
   153.0 bits = 1.117 bpc x 137 chars
   285.5 bits = 1.310 bpc x 218 chars
   147.0 bits = 1.050 bpc x 140 chars
    90.7 bits = 0.749 bpc x 121 chars
   413.1 bits = 1.700 bpc x 243 chars
   299.1 bits = 1.262 bpc x 237 chars
   292.1 bits = 1.137 bpc x 257 chars
   515.9 bits = 1.664 bpc x 310 chars
   298.1 bits = 1.666 bpc x 179 chars
   200.0 bits = 1.399 bpc x 143 chars
```

```
macpro:~/lua/projects/char-rnn > make test_wine_red
head -n 10 data/wine_red/wine_red_test.txt | th score.lua  wine_red/
    76.1 bits = 0.556 bpc x 137 chars
   210.1 bits = 0.964 bpc x 218 chars
   135.5 bits = 0.968 bpc x 140 chars
    91.6 bits = 0.757 bpc x 121 chars
   221.3 bits = 0.911 bpc x 243 chars
   179.9 bits = 0.759 bpc x 237 chars
   197.7 bits = 0.769 bpc x 257 chars
   263.9 bits = 0.851 bpc x 310 chars
   131.0 bits = 0.732 bpc x 179 chars
   102.2 bits = 0.715 bpc x 143 chars
```

Ability to compress = log-likelihood

High compression = good match

Wharton
Department of Statistics

Closing Comments

Parting Comments

Text analytics

Continues to move into mainstream

Objectives

Build features for "familiar" models
Understanding the structure of language

Issues of statistical modeling for large data sets remain

Overfitting, missing data, outliers, …

Computing

Methods related to deep learning have become more widely accessible, and hence more common

What's the role for the social scientist?