Text Processing: Overview

-Quantifying the World-

Lecturer: Darren Homrighausen, PhD

Who was the first pope?

Suppose we are having a bar-room debate with our friends about the origins of the papacy

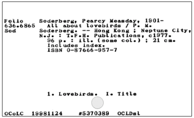
How we would settle this debate has changed radically in the last 20 years.

What we used to do

1. Go to library

2. Card catalog

3. Get metadata



Folio 636.6865 Sod
Soderberg, Pearcy Measday, 1901-
All about lovebirds / P. M. Soderberg. -- Hong Kong ; Neptune City, N.J. : T.F.H. Publications, c1977.
96 p. : ill. (some col.) ; 21 cm.
Includes index.
ISBN 0-87666-957-7

1. Lovebirds. I. Title

OCoLC 19981124 #5370389 OCLDsl

4. Search

5. No book

6. Wait

This was slow and expensive..

What we do now

Information retrieval and representations

How does Google do this?

Information retrieval: given a set of documents (such as webpages, emails, news articles,..), our problem is to retrieve the K most similar documents to a given query (e.g. "who was the first pope?").

The first step is to think of a way of representing these documents.

We want the representation to:

- be both easy to generate from the documents and easy to work with
- highlight important aspects of the documents and suppress unimportant ones

Like always, there is a trade-off between these two ideas

Bag-of-words representation

It turns out a very simple minded approach is probably the best developed so far. Take all the words in the document(s) and count how many times they appear and stick this in a long vector (or matrix, if multiple documents).

(An extension of this idea is to *n*-grams which are sections of text)

For example:

```
pope = 154, catholic = 17, vatican = 12, jesus = 2, the = 304,..
```

This is very easy to generate (once we tweak the scripting to ignore certain things).

But is it too much of a reduction?

Bag-of-words representation

Idea: By itself “pope” can mean different things

But, we can learn from the other words in the document

- Words like ‘football’, ‘NFL’, ‘lineman’, and ‘arizona’ suggest the wrong type of pope
- Words like ‘pontiff’, ‘vatican’, ‘catholic’, and ‘italy’ suggest the right type of pope
- Words like ‘cardinal’ are not informative

Counting words

Recall problem: given a query and a set of documents, find the K documents most similar to the query

Countings words:

1. Make a list of all the words present in the documents and the query
2. Index the words $w = 1, \ldots, W$
   (for example, in alphabetical order)
3. Index the documents $d = 1, \ldots, D$
   (just pick some order)
4. For each document d, count how many times each word w is used (can be, and most likely is, zero), and call this count X_{dw}. The vector $X_d = (X_{d1}, \ldots, X_{dW})^\top$ gives the word counts for the d^{th} document
5. Lastly, do the same thing for the query $Y = (Y_1, \ldots, Y_W)^\top$ and Y_w is the count for word w in the query

Simple example

Documents:

$d = 1$: "This statistics class is classy'
$d = 2$: "statistics say this statistics class has no class'

Query:

"classy statistics class"

	this	statistics	class	classy	is	has	no	say
X_1	1	1	1	1	1	0	0	0
X_2	1	2	2	0	0	1	1	1
Y	0	1	1	1	0	0	0	0

This is known as a document-term matrix

Distances and similarity measures

We represented each document X_d and query Y in a convenient vector format. Now, how do we measure similarity?

Measures of distance between W-dimensional vectors X and Y:

- The ℓ_2 or Euclidean distance is

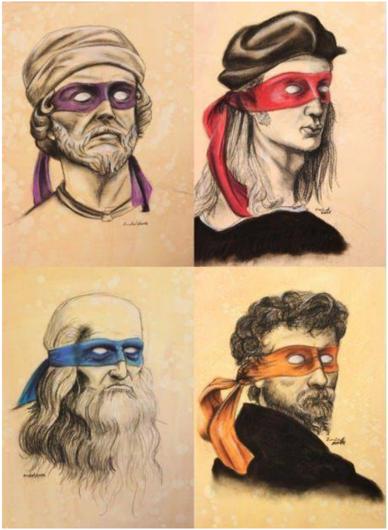
$$||X - Y||_2 = \sqrt{\sum_{w=1}^{W} (X_w - Y_w)^2}$$

- The ℓ_1 or Manhattan distance is

$$||X - Y||_2 = \sum_{w=1}^{W} |X_w - Y_w|$$

There are many others

Basic idea: Find the K vectors X that are 'closest'



Bigger example

Documents: Suppose we have 8 Wikipedia articles, 4 about the TMNT (Leonardo, Raphael, Michelangelo, and Donatello) and about the 4 renaissance artists of the same name

1

2

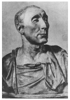
3

4

5

6

7



8

Query: "Raphael is cool but rude, Michelangelo is a party dude!"

Potential problems

What are the potential problems with performing this query?

Potential problems

What are the potential problems with performing this query?

- Unequal document sizes (example: TMNT Michelangel is 3330 words vs 6524 words for the artist). Also, the query is only 7 words long.
- Stemming
- Punctuation
- Common words ('raphael' occurs 70 times in TMNT article and 144 in the artist's article) provide little discrimination

Distances

If we don't account for any of these problems, we get the following subset of the document-term matrix along with the distance to the query

	but	cool	dude	party	michelangelo	raphael	rude	dist
doc 1	19	0	0	0	4	24	0	309.5
doc 2	8	1	0	0	7	45	1	185.2
doc 3	7	0	4	3	77	23	0	331.0
doc 4	2	0	0	0	4	11	0	220.2
doc 5	17	0	0	0	9	6	0	928.5
doc 6	36	0	0	0	17	101	0	646.5
doc 7	10	0	0	0	159	2	0	527.3
doc 8	2	0	0	0	0	0	0	196.1
query	1	1	1	1	1	1	1	0.0

1. Raphael the Turtle
2. Donatello the Artist
3. Michelangelo the Turtle

Distances

If we don't account for any of these problems, we get the following subset of the document-term matrix along with the distance to the query

	but	cool	dude	party	michelangelo	raphael	rude	dist
doc 1	19	0	0	0	4	24	0	309.5
doc 2	8	1	0	0	7	45	1	185.2
doc 3	7	0	4	3	77	23	0	331.0
doc 4	2	0	0	0	4	11	0	220.2
doc 5	17	0	0	0	9	6	0	928.5
doc 6	36	0	0	0	17	101	0	646.5
doc 7	10	0	0	0	159	2	0	527.3
doc 8	2	0	0	0	0	0	0	196.1
query	1	1	1	1	1	1	1	0.0

1. Raphael the Turtle
2. Donatello the Artist
3. Michelangelo the Turtle

Varying document lengths and normalization

Different documents have different lengths. Total word counts:

doc 1	doc 2	doc 3	doc 4	doc 5	doc 6	doc 7	doc 8	query
3114	1976	3330	2143	8962	6524	4618	1766	7

Note that the documents have quite different lengths

We should normalize them in some way

- Document length: Divide X by its sum

$$X \leftarrow X / \sum_{w=1}^{W} X_w$$

- ℓ_2 length: Divide X by its Euclidean length

$$X \leftarrow X / ||X||_2$$

Back to our example

```
    dist/doclen  dist/l2len
doc 1 0.3852650 1.373041
doc 2 0.3777634 1.321871
doc 3 0.3781194 1.319048
doc 4 0.3887862 1.393433
doc 5 0.3906030 1.404972
doc 6 0.3820197 1.349070
doc 7 0.3812202 1.324758
doc 8 0.3935327 1.411486
query 0.0000000 0.000000
```

Great!

So far, we've dealt with the varying document lengths. What about some words being more helpful than others?

Common words, especially, are not going to help find relevant documents

How do we deal with common words?

Intuition: Words that do not appear very often should help us discriminate better, as they are by implication very specific to that document

To deal with common words we could just keep track of a list of words like ‘the’, ‘this’, ‘that’, etc. and exclude them from our representation.

This is both too crude and time consuming

Common words and IDF weighting

Inverse document frequency (IDF) weighting is smarter and more efficient

- For each word, w, let n_w be the number of documents that contain this word
- The, for each vector X_d and Y, multiply the w^{th} component by
$$IDF(w) = \log(D/n_w)$$

Note that if a word appears in every document, then it gets a weight of zero.

If $n_w < D$, then $\log(D/n_w) > 0$. In particular, if $D >> n_w$, then D/n_w is also large (example: $D = 100$, $n_w = 1$, $IDF(w) \approx 4.6$)

Putting it all together

Think of the document-term matrix

	word 1	word 2	...	word W
doc 1				
doc 2				
$\vdots$				
doc D				

- Normalization scales each *row* by something (divides a row vector X by its sum or ℓ_2 norm)
- IDF weighting scales each *column* by something (multiplies the w^{th} column by $IDF(w)$)
- We can use both, just normalize first and then perform IDF

Back to our example

```
                          dist/doclen/IDF
doc 1 (tmnt leo)                    0.623
doc 2 (tmnt rap)                    0.622
doc 3 (tmnt mic)                    0.620
doc 4 (tmnt don)                    0.623
doc 5 (real leo)                    0.622
doc 6 (real rap)                    0.622
doc 7 (real mic)                    0.622
doc 8 (real don)                    0.624
query (tmnt leo)                    0.000
```

What happened?

```
 [1] "---"         "----"        "---x"        "--dead"      "--x"         "-foot"
 [7] "-part"       "-year-old"   "abandoned"   "abbeville"   "abbey"       "abilities"
[13] "ability"     "able"        "abode"       "about"       "above"       "abrams"
[19] "abroad"      "abruptly"    "absence"     "absent"      "absolute"    "absorbed"
[25] "absorbing"   "abstemious"  "absurd"      "abundance"   "abundantly"  "abuse"
[31] "academic"    "academies"   "academy"     "accademia"   "accent"      "accept"
[37] "acceptance"  "accepted"    "accepting"   "accident"    "acclaimed"   "acclimate"
[43] "accompany"   "accomplish"  "accordance"  "according"   "account"     "accounts"
```

Stemming

Having words ‘connect’, ‘connects’, ‘connected’, ‘connecting’, ‘connection’, etc. in our representation is extraneous. Stemming reduces all of these to a single stem word ‘connect’

Can a simple list of rules provide perfect stemming? No: ‘relate’ vs. ‘relativity’ or ‘sand’ and ‘sander’ or ‘wand’ and ‘wander’ or ‘man’ and ‘many’ or...

Stemming also depends on the language. It is easier in English than in:

- German (fusional or agglomerative language) e.g.
  Hubschrauberlandeplatz = helicopter landing pad
- Turkisk (agglutinative language) e.g.
  Turklestiremedigimizlerdensinizdir = maybe you are one of those whom we were not able to Turkify

Stemming

Before

```
 [1] "---"         "----"        "---x"        "--dead"      "--x"         "-foot"
 [7] "-part"       "-year-old"   "abandoned"   "abbeville"   "abbey"       "abilities"
[13] "ability"     "able"        "abode"       "about"       "above"       "abrams"
[19] "abroad"      "abruptly"    "absence"     "absent"      "absolute"    "absorbed"
[25] "absorbing"   "abstemious"  "absurd"      "abundance"   "abundantly"  "abuse"
[31] "academic"    "academies"   "academy"     "accademia"   "accent"      "accept"
[37] "acceptance"  "accepted"    "accepting"   "accident"    "acclaimed"   "acclimate"
[43] "accompany"   "accomplish"  "accordance"  "according"   "account"     "accounts"
```

After

```
 [1] "---"         "----"        "---x"        "--dead"      "--x"         "-foot"
 [7] "-part"       "-year-old"   "abandon"     "abbevill"    "abbey"       "abil"
[13] "abl"         "abod"        "abov"        "abram"       "abroad"      "abrupt"
[19] "absenc"      "absent"      "absolut"     "absorb"      "abstemi"     "absurd"
[25] "abund"       "abus"        "academ"      "academi"     "accademia"   "accent"
[31] "accept"      "accid"       "acclaim"     "acclim"      "accommod"    "accompani"
[37] "accomplish"  "accord"      "account"     "accumul"     "accur"       "accus"
[43] "achiev"      "ackerman"    "acknowledg"  "acolyt"      "acquir"      "act"
```

Back to our example, after stemming


1

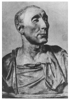
2

3

4

5

6

7

8

Query: "Raphael is cool but rude, Michelangelo is a party dude!"

	dist/doclen/IDF
doc 1 (tmnt leo)	0.965
doc 2 (tmnt rap)	0.870
doc 3 (tmnt mic)	0.867
doc 4 (tmnt don)	0.971
doc 5 (real leo)	0.927
doc 6 (real rap)	0.971
doc 7 (real mic)	0.954
doc 8 (real don)	0.930
query	0.000

Text processing in R

Text processing in R

The basic commands are:

```
library(tm)
corp = VCorpus(VectorSource(docs))
dtm  = DocumentTermMatrix(corp,
                          control=list(tolower=TRUE,
                          removePunctuation=TRUE,
                          removeNumbers=TRUE))
```

For example:

```
exampleDoc = c('I really want a real pony not a wanted poster')
exampleCorp = VCorpus(VectorSource(exampleDoc))
dtm = DocumentTermMatrix(exampleCorp,
                        control=list(tolower=TRUE,
                        removePunctuation=TRUE,
                        removeNumbers=TRUE))
```

```
> colnames(dtm)
[1] "not"  "pony"  "poster" "real"  "really" "want"   "wanted"
```

Text processing in R

```
exampleDoc1 = c('I really want a real pony not a wanted poster')
exampleDoc2 = c('Real men do not ride ponies, they ride rockets'
exampleDoc3 = c('I had a pony named rocket, man')
exampleDocs = c(exampleDoc1,exampleDoc2,exampleDoc3)
exampleCorp = VCorpus(VectorSource(exampleDocs))
dtm = DocumentTermMatrix(exampleCorp,
                         control=list(tolower=TRUE,
                         removePunctuation=TRUE,
                         removeNumbers=TRUE))
> colnames(dtm)
 [1] "had"   "man"    "men"     "named"   "not"     "ponies"
"pony"      "poster"  "real"    "really"  "ride"    "rocket"
"rockets" "they"   "want"   "wanted"
> dtm
A document-term matrix (3 documents, 16 terms)
Non-/sparse entries: 19/29
Sparsity           : 60%
Maximal term length: 7
Weighting          : term frequency (tf)
```

Why sparsity?

Reminder: Sparse matrix structures can be really helpful

In text processing, sparsity is everything

I have a dataset with approximately 30,000 words and 52,000 documents. If I stored this naively, this would take

$$\text{storage} = \frac{64\text{bits} * 30,000 * 52,000}{8\text{bytes} * 2^{10}\text{kb} * 2^{10}\text{mb} * 2^{10}\text{gigabytes}} = 11.622\text{gb}$$

Text processing in R

We can look directly at the document-term matrix

```
> inspect(dtm)
[omitted]
had man men named not ponies pony poster real really ride rocket
0   0   0     0   1      0    1      1    1      1    0      0
0   0   1     0   1      1    0      0    1      0    2      0
1   1   0     1   0      0    1      0    0      0    0      1

rockets they want  wanted
0        0    1        1
1        1    0        0
0        0    0        0
```

Text processing in R

Reminder, here is our index ($W = 16$)

```
> colnames(dtm)
 [1] "had"   "man"    "men"     "named"   "not"     "ponies"
"pony"      "poster"  "real"  "really"  "ride"    "rocket"
 "rockets" "they"   "want"   "wanted"
```

There are two issues here: common words and stemming. We can with both relatively easily in R

Dealing with common words and stemming

```
> dtm.stem = DocumentTermMatrix(exampleCorp,
          control=list(tolower=TRUE,
          removePunctuation=list(preserve_intra_word_dashes=T),
          removeNumbers=TRUE,
          stemming=TRUE,
          stopwords = TRUE,
          weighting=weightTfIdf,
          wordLengths = c(3,10))
```

- `removePunctuation`: Here, I have it keep between word dashes to maintain hyphenation
- `stemming`: Should I perform stemming?
- `stopwords`: These are common transition words that are called stop words. These are words like 'the', 'at', 'a' ...
- `weighting`: What weighting scheme should I do?
- `wordLengths`: What length of words should I accept?

Text processing in R: New dictionaries

Reminder, here is our index ($W = 16$)

```
> colnames(dtm)
"had"   "man"    "men"      "named"    "not"      "ponies"
"pony"    "poster"  "real"   "really"  "ride"    "rocket"   "rockets"
"they"   "want"   "wanted"
> colnames(dtm.stem)
"name"    "poni"    "poster" "real"
"realli" "ride"    "rocket"
```

What happens if I don't remove stop words?
(set `stopwords = FALSE`)

```
> colnames(dtm.nostop)
 [1] "do"    "had"   "man"   "men"   "name"    "not"   "poni"
 "poster" "real" "realli" "ride"    "rocket" "they" "want"
```

Text processing in R: Inputs to Distances

```
> inspect(dtm.stem)
A document-term matrix (3 documents, 7 terms)

Non-/sparse entries: 8/13
Sparsity           : 62%
Maximal term length: 6
Weighting          : term frequency - inverse document frequency
(normalized) (tf-idf)

    Terms
    name       poni   poster       real      realli      ride   rocket
1 0.0000000    0.0 0.3962406 0.1462406 0.3962406 0.000000 0.000
2 0.0000000    0.0 0.0000000 0.1169925 0.0000000 0.633985 0.116
3 0.5283208    0.0 0.0000000 0.0000000 0.0000000 0.000000 0.194

# So, Doc 1 and Doc 2 are
> mydtm = as.matrix(dtm.stem)
> sqrt(sum((mydtm[1,]-mydtm[2,])^2))
[1] 0.8546888
```

Text processing in R: Hazards of open-source software

You may need to do the following at the beginning of your code:

```
Sys.setenv(NOAWT=TRUE)
library(RWeka)
library(rJava)
```

Note: install these packages first

Latent Semantic Analysis

PCA and factor analysis interpretation

- PCA: Find the directions of greatest variance. This doesn't on its face seem like it maintains correlations, but observe:

  $$var(aX_1 + bX_2) = a^2 Var(X_1) + b^2 Var(X_2) + 2abCov(X_1, X_2)$$

  If we standardize the matrix, then this reduces to

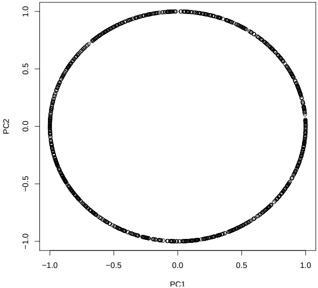
  $$var(aX_1 + bX_2) = a^2 + b^2 + 2abCov(X_1, X_2)$$

  This gets maximized over $a^2 + b^2 = 1$.
  - If $Cov(X_1, X_2) \approx 0$, then this gets maximized by any $a^2 + b^2 = 1$ (it doesn't matter)
  - If $Cov(X_1, X_2) \approx 1$, then this gets maximized by setting $a = b = 1/\sqrt{2}$
- Factor analysis: Defined by maintaining correlations.

So, in either case, we are really maintaining correlations

Graphical example of the phenomenon

```
library(mvtnorm)
sigma   = matrix(c(1,sig,sig,1),nrow=2)
nsweep  = 1000
outcome = matrix(0,nrow=nsweep,ncol=2)
for(sweep in 1:nsweep){
x                = rmvnorm(200,c(0,0),sigma)
out.pca          = prcomp(x,center=T,scale=F)
outcome[sweep,] = out.pca$rotation[,1]
}
plot(outcome,xlab='PC1',ylab='PC2')
```



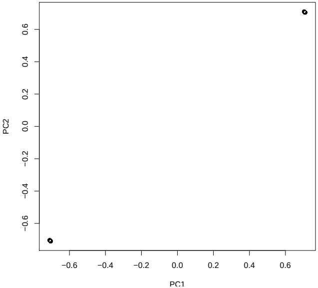


Figure: Left: sig = 0. Right: sig = .999

How does this apply to text processing?

Think about the document-term matrix $\mathbb{X}$.

The columns correspond to the words. If two words w_1, w_2 commonly appear together then the w_1^{th} and w_2^{th} columns of $\mathbb{X}$ are correlated

When we have a large number of documents that are about a topic, it is common to have some, or most, of the documents using related, but not identical words

Therefore, if we were to search $\mathbb{X}$ with a query, we would miss some of the important documents

An example

Suppose we have a corpus of documents

We wish to search for documents containing agriculture

We can query $Y =$ ("agriculture")

However, "agriculture" is not regularly explicitly mentioned in articles about agriculture

This is where correlations come in. Whenever agriculture is mentioned, it will occur very frequently along with many synonyms ("farming", for instance)

This is where latent semantic indexing comes in

An example, given to us by an invisible hand

To see why it is called latent semantic *indexing*, observe the following

When a book is written, a list of terms (or topics) is written down and an index is formed saying where these terms appear. For example, here is the start to the entry for "Agriculture" in the index to *The Wealth of Nations*

An example, given to us by an invisible hand

It is asking a lot for a computer to do this.

However, if we only want to get the pages where "agriculture" is the topic (like, 6, 92, 152–3, 220..), then we can make a document-term matrix out of the pages of the book.

This approach will fail if we search this document-term matrix directly.

However, asking for pages that contain highly correlated words (like "rent") should work very well

Latent semantic indexing (LSI)

If we have our document-term matrix $\mathbb{X}$, then we write $\mathbb{X} = UDV^\top$, where

- The matrix U is the document-concept matrix
  (The columns are the documents in concept space)
- The matrix V is the term-concept matrix
  (The columns are the terms in concept space)

If we have our query Y, we can map it into the document space by thinking of it as a new row in $\mathbb{X}$

$$\mathbb{X} = UDV^\top \quad \text{is the same as} \quad \mathbb{X}VD^{-1} = U$$

Which means we transform Y as

$$YVD^{-1}$$

To center or not to center?

If we think about LSI as performing PCA, then we should technically do the SVD like

$$\mathbb{X} - \overline{\mathbb{X}} = UDV^\top$$

However, observe the following example

$$A = \begin{bmatrix} a & b & 0 \\ d & 0 & f \\ 0 & h & i \end{bmatrix}$$

What happens if we column center A?

To center or not to center?

If we think about LSI as performing PCA, then we should technically do the SVD like

$$\mathbb{X} - \overline{\mathbb{X}} = UDV^\top$$

However, observe the following example

$$A = \begin{bmatrix} a & b & 0 \\ d & 0 & f \\ 0 & h & i \end{bmatrix}$$

What happens if we column center A?

We lose sparsity!

To center or not to center?

If we think about LSI as performing PCA, then we should technically do the SVD like

$$\mathbb{X} - \overline{\mathbb{X}} = UDV^\top$$

However, observe the following example

$$A = \begin{bmatrix} a & b & 0 \\ d & 0 & f \\ 0 & h & i \end{bmatrix}$$

What happens if we column center A?

We lose sparsity!

The consensus is to column center if your data is small enough, otherwise, don't worry about it

Example dataset

Let's look at $D = 20$ Reuters news articles about crude oil production and importation.

The corpus has 860 words

```
"Diamond Shamrock Corp said that\neffective today it
had cut its contract prices for crude oil by\n1.50 dlrs a
barrel.\n    The reduction brings its posted price for
West Texas\nIntermediate to 16.00 dlrs a barrel, the
copany said.\n    \"The price reduction today was made
in the light of falling\noil product prices and a weak crude
oil market,\" a company\nspokeswoman said.\n    Diamond
is the latest in a line of U.S. oil companies that\nhave cut
its contract, or posted, prices over the last two days\nciting
weak oil markets.\n Reuter"
```

Example dataset: R

Let's look at

```
# Dense solver
lsi_dense = svd(docTermSparse)

# Sparse solver
require(irlba)
lsi_sparse = irlba(docTermSparse,
                   nv = 10,nu = 10,
                   tol=1e-10)

termConcept = lsi_sparse$v
signif(sort(termConcept[,1],decreasing=TRUE)[1:24],2)
signif(sort(termConcept[,1],decreasing=FALSE)[1:24],2)
```

Term concepts for Reuters documents

```
> signif(sort(termConcept[,1],decreasing=TRUE)[1:24],2)
    january       cubic    fiscales petroliferos  yacimientos      billion
     0.5000      0.3200      0.3200       0.3200       0.3200       0.2500
  argentine         gas     metrers       metres      natural     produced
     0.1600      0.1600      0.1600       0.1600       0.1600       0.1600
   totalled     barrels       added   production          pct          mln
     0.1600      0.0940      0.0860       0.0850       0.0780       0.0770
     output      budget      riyals   abdul-aziz  expenditure      revenue
     0.0630      0.0061      0.0061       0.0051       0.0051       0.0041
> signif(sort(termConcept[,1],decreasing=FALSE)[1:24],2)
  posted    canada  canadian      west     power       bbl   lowered    texaco
  -0.050    -0.044    -0.044    -0.041    -0.040    -0.040    -0.036    -0.033
   texas    brings effective      dlrs     grade     sweet  contract      ship
  -0.033    -0.032    -0.032    -0.031    -0.031    -0.031    -0.031    -0.030
   price   changed       pay  postings  decrease   company benchmark       feb
  -0.030    -0.028    -0.028    -0.028    -0.027    -0.027    -0.026    -0.026
```

These are the 24 largest and smallest values in the first column of term-concept matrix (V[,1])

What story can be told here?

Term concepts for Reuters documents

```
> signif(sort(termConcept[,1],decreasing=TRUE)[1:24],2)
    january        cubic     fiscales petroliferos  yacimientos      billion
     0.5000       0.3200       0.3200       0.3200       0.3200       0.2500
  argentine          gas      metrers       metres      natural     produced
     0.1600       0.1600       0.1600       0.1600       0.1600       0.1600
   totalled      barrels        added   production          pct          mln
     0.1600       0.0940       0.0860       0.0850       0.0780       0.0770
     output       budget       riyals   abdul-aziz  expenditure      revenue
     0.0630       0.0061       0.0061       0.0051       0.0051       0.0041
> signif(sort(termConcept[,1],decreasing=FALSE)[1:24],2)
   posted    canada  canadian      west     power       bbl   lowered    texaco
   -0.050    -0.044    -0.044    -0.041    -0.040    -0.040    -0.036    -0.033
    texas    brings effective      dlrs     grade     sweet  contract      ship
   -0.033    -0.032    -0.032    -0.031    -0.031    -0.031    -0.031    -0.030
    price   changed       pay  postings  decrease   company benchmark       feb
   -0.030    -0.028    -0.028    -0.028    -0.027    -0.027    -0.026    -0.026
```

These are the 24 largest and smallest values in the first column of term-concept matrix (V[,1])

What story can be told here?

- Large loadings correspond to things related to the international market
- Negative loadings correspond to the American/Canadian market

Term concepts for TMNT documents

```
> signif(sort(termConcept[,1],decreasing=TRUE)[1:24],2)
     cool       rude       dude      parti    raphael      -foot        --x
  5.0e-01    5.0e-01    5.0e-01    5.0e-01    3.6e-02   -8.1e-05   -8.1e-05
  acclaim acknowledg        add      adequ   administ     aerial      ahead
 -8.1e-05   -8.1e-05   -8.1e-05   -8.1e-05   -8.1e-05   -8.1e-05   -8.1e-05
  albiera  albrecht alessandra       amaz ambrosiana     amount     analys
 -8.1e-05   -8.1e-05   -8.1e-05   -8.1e-05   -8.1e-05   -8.1e-05   -8.1e-05
> signif(sort(termConcept[,1],decreasing=FALSE)[1:24],2)
    turtl      comic donatello      paint      ninja    florenc   leonardo     mutant
  -0.0180    -0.0100   -0.0085    -0.0081    -0.0076    -0.0073    -0.0070    -0.0063
     seri      statu       art       voic       anim       tmnt       game    brother
  -0.0061    -0.0047   -0.0047    -0.0045    -0.0044    -0.0043    -0.0039    -0.0038
  charact   shredder     duomo     medici      casey   splinter     teenag      penni
  -0.0037    -0.0036   -0.0035    -0.0034    -0.0032    -0.0032    -0.0032    -0.0032
```

These are the 24 largest and smallest values in the first column of term-concept matrix (V[,1])

What story can be told here?

Term concepts for TMNT documents

```
> signif(sort(termConcept[,1],decreasing=TRUE)[1:24],2)
     cool       rude       dude      parti    raphael      -foot        --x 
  5.0e-01    5.0e-01    5.0e-01    5.0e-01    3.6e-02   -8.1e-05   -8.1e-05 
  acclaim acknowledg        add      adequ   administ     aerial      ahead 
 -8.1e-05   -8.1e-05   -8.1e-05   -8.1e-05   -8.1e-05   -8.1e-05   -8.1e-05 
  albiera  albrecht alessandra       amaz ambrosiana     amount     analys 
 -8.1e-05   -8.1e-05   -8.1e-05   -8.1e-05   -8.1e-05   -8.1e-05   -8.1e-05 
> signif(sort(termConcept[,1],decreasing=FALSE)[1:24],2)
    turtl     comic donatello     paint     ninja   florenc  leonardo    mutant 
  -0.0180   -0.0100   -0.0085   -0.0081   -0.0076   -0.0073   -0.0070   -0.0063 
     seri     statu       art      voic      anim      tmnt      game   brother 
  -0.0061   -0.0047   -0.0047   -0.0045   -0.0044   -0.0043   -0.0039   -0.0038 
  charact  shredder     duomo    medici     casey  splinter    teenag     penni 
  -0.0037   -0.0036   -0.0035   -0.0034   -0.0032   -0.0032   -0.0032   -0.0032 
```

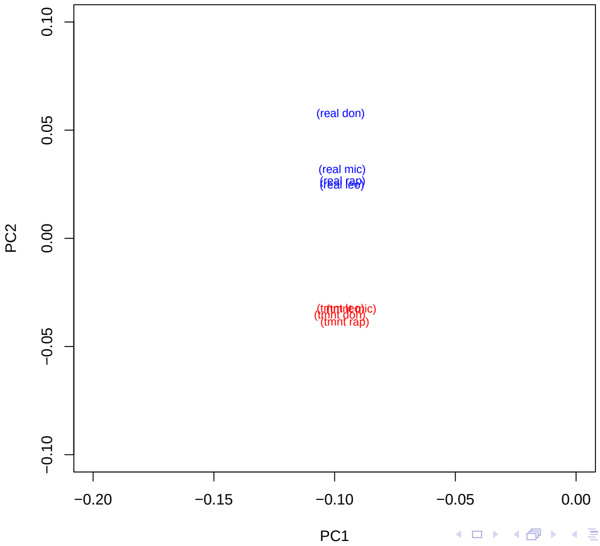
These are the 24 largest and smallest values in the first column of term-concept matrix (V[,1])

What story can be told here?

- There is a substantial amount of overlap on the first concept

Let's look at a plot

Plot of PC scores for TMNT



Term concepts for TMNT documents

```
> signif(sort(termConcept[,2],decreasing=TRUE)[1:24],2)
  florenc      statu      paint      duomo        art   basilica     medici
    0.220      0.150      0.140      0.130      0.110      0.083      0.080
    museo equestrian    madonna     execut        del      bronz     firenz
    0.074      0.073      0.071      0.070      0.069      0.063      0.062
   cosimo     judith    lorenzo    prophet      penni     vasari    centuri
    0.059      0.059      0.057      0.057      0.055      0.054      0.054
> signif(sort(termConcept[,2],decreasing=FALSE)[1:24],2)
  turtl    comic    ninja   mutant     seri     voic     anim     tmnt     game
 -0.420   -0.240   -0.180   -0.150   -0.140   -0.100   -0.100   -0.100   -0.092
charact  brother shredder    casey   teenag splinter     foot      sai    mirag
 -0.087   -0.083   -0.081   -0.077   -0.074   -0.074   -0.067   -0.066   -0.065
   movi    mikey   episod    mutat     clan     raph
 -0.063   -0.061   -0.056   -0.049   -0.046   -0.045
```

These are the 24 largest and smallest values in the first column of term-concept matrix (V[,2])

What story can be told here?

Term concepts for TMNT documents

```
> signif(sort(termConcept[,2],decreasing=TRUE)[1:24],2)
  florenc       statu       paint       duomo         art    basilica      medici
    0.220       0.150       0.140       0.130       0.110       0.083       0.080
    museo equestrian     madonna      execut         del       bronz      firenz
    0.074       0.073       0.071       0.070       0.069       0.063       0.062
   cosimo      judith     lorenzo     prophet       penni      vasari     centuri
    0.059       0.059       0.057       0.057       0.055       0.054       0.054
> signif(sort(termConcept[,2],decreasing=FALSE)[1:24],2)
  turtl    comic    ninja   mutant     seri     voic     anim     tmnt     game
 -0.420   -0.240   -0.180   -0.150   -0.140   -0.100   -0.100   -0.100   -0.092
charact  brother shredder    casey   teenag splinter     foot      sai    mirag
 -0.087   -0.083   -0.081   -0.077   -0.074   -0.074   -0.067   -0.066   -0.065
   movi    mikey   episod    mutat     clan     raph
 -0.063   -0.061   -0.056   -0.049   -0.046   -0.045
```

These are the 24 largest and smallest values in the first column of term-concept matrix (V[,2])

What story can be told here?

- Positive values are related to the renaissance artists
- Negative values are related to the TMNTs

Distance to query using LSI

Original representation

[dimension reduction]

Statistical method

- Form the (normalized) document-term matrix $\mathbb{X}$
- Compute its LSI $\mathbb{X} = UDV^\top$
- Get Y into LSI via $\tilde{Y} = YVD^{-1}$
- Choose a K
- Find distances for documents $d = 1, \ldots, D$

$$\mathrm{distance}(d, \tilde{Y}) = ||U[d, 1:K] - \tilde{Y}||_2$$

(The document-concept matrix restricted to the first K columns)

Distance to query using LSI

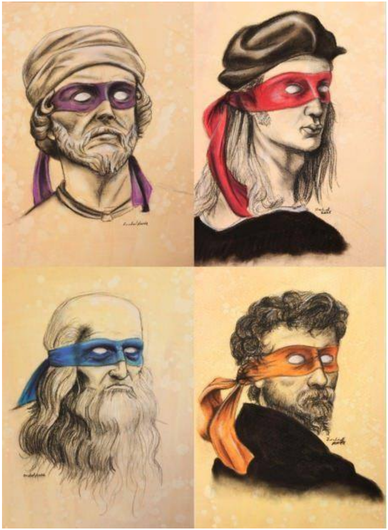
```
(tmnt leo) 0.9711807
(tmnt rap) 0.7696525
(tmnt mic) 0.7669749
(tmnt don) 0.9718710
(real leo) 0.9711512
(real rap) 0.9709391
(real mic) 0.9709492
(real don) 0.9734319
query      0.0000000
```

Supervised Methods

Making a new data set

Let's make a new data set that breaks the TMNT docs into pieces

We can use it review some of the previously considered methods



Making a new data set

The main parts of the R code

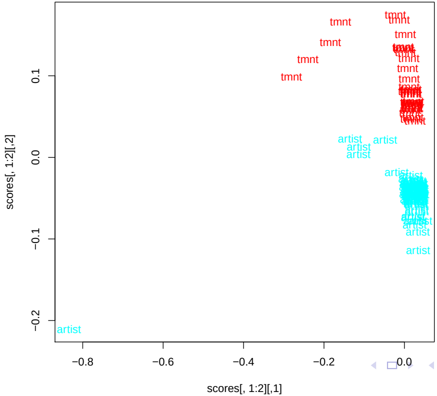
```
nCharInDoc = 2500
splitDocs = c()
for(doc in docs){
  notAtEnd = T
  iterSplit = 0
  while(notAtEnd){
    iterSplit = iterSplit + 1
    newChunk = substr(doc,(iterSplit - 1)*nCharInDoc+1,
                      iterSplit*nCharInDoc)
    splitDocs = c(splitDocs,newChunk)
    if(nchar(newChunk) < nCharInDoc){
      notAtEnd = F
    }
  }
}
```

This generates a doc-term matrix that is 109×4778

Document-Concept

Let's look at the documents plotted in terms of the concepts:

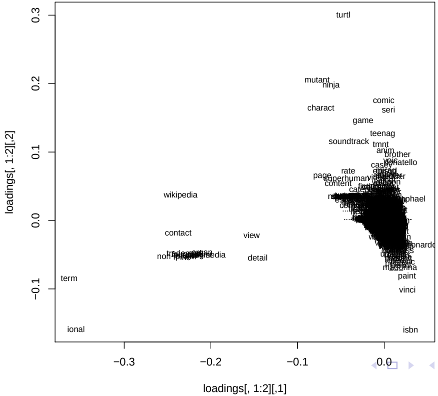
```
out.lsi = svd(mydtm)
docConcept   = out.lsi$u
plot(docConcept[,1:2],type='n')
text(docConcept[,1:2],label=Y,col=color)
```



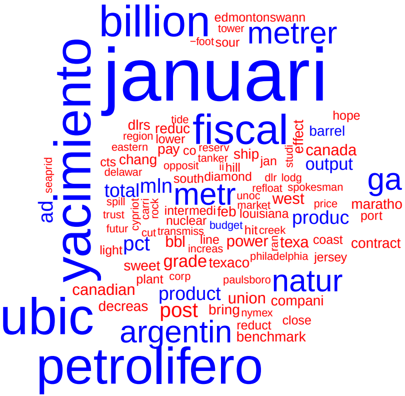
Term-Concept

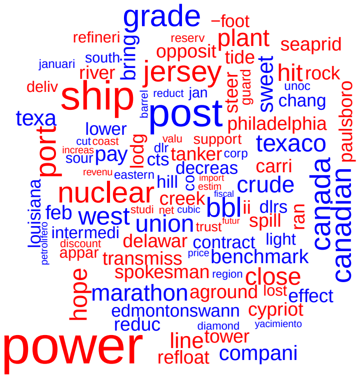
Let's look at the loadings:

```
out.lsi = svd(mydtm)
termConcept    = out.lsi$v
plot(termConcept[,1:2],type='n')
text(termConcept[,1:2],label=colnames(mydtm),cex=.5)
```



Word clouds for LSI: Routers articles using color





abs(termConcept[,1]): positive, negative

abs(termConcept[,2]): positive, negative

ℓ_1-REGULARIZED REGRESSION: REMINDER

Known as

- 'lasso'
- 'basis pursuit'

The estimator satisfies

$$\hat{\beta}_{lasso}(t) = \underset{||\beta||_1 \leq t}{\operatorname{argmin}} ||\mathbb{Y} - \mathbb{X}\beta||_2^2$$

In its corresponding Lagrangian dual form:

$$\hat{\beta}_{lasso}(\lambda) = \underset{\beta}{\operatorname{argmin}} ||\mathbb{Y} - \mathbb{X}\beta||_2^2 + \lambda||\beta||_1$$

Generalized linear models: Reminder

GLMs differ from ordinary regression by modeling the *probabilities* as opposed to the outcomes themselves. To wit:

Regression:

$$Y_i = X_i^\top \beta + \epsilon_i$$

Logistic regression (with logit link): Let $\pi(X_i) = Pr(Y_i = 1|X_i)$,

$$\log\left(\frac{\pi(X_i)}{1-\pi(X_i)}\right) = X_i^\top \beta$$

This is known as the logistic function.

It is differentiable, maps [0,1] to $\mathbb{R}$, and is invertible. Its inverse is:

$$\pi(X_i) = \frac{\exp\{X_i^\top \beta\}}{1+\exp\{X_i^\top \beta\}}.$$

Combining lasso with GLMs

If you've been wondering where the glm part of glmnet comes from, today is your day

It turns out that this definition of the lasso:

$$\hat{\beta}_{lasso}(\lambda) = \underset{\beta}{\text{argmin}} \, ||\mathbb{Y} - \mathbb{X}\beta||_2^2 + \lambda||\beta||_1$$

can be viewed as assuming a Gaussian likelihood

Maximum likelihood estimation

Suppose we have data $(X_1, Y_1), \ldots, (X_n, Y_n)$

Assume that $Y_i | X_i \sim N(X_i^\top \beta, \sigma^2)$
(This is the usually multiple regression assumption using a concise notation)

We can write its likelihood for any β as:

$$L(\beta) = \frac{1}{(2\pi\sigma^2)^{n/2}} e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(Y_i - X_i^\top \beta)^2}$$

If we wanted to find the maximum likelihood estimator of β, we would find max $L(\beta)$

Maximum likelihood estimation

The maximizer of $L(\beta)$ is the same as $\ell(\beta) = \log L(\beta)$.

$$\begin{aligned}
\ell(\beta) &= -\frac{n}{2}\left(\log(2\pi) + \log(\sigma^2)\right) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(Y_i - X_i^\top \beta)^2 \\
&\propto -\sum_{i=1}^{n}(Y_i - X_i^\top \beta)^2 \\
&= -||Y - \mathbb{X}\,\beta||_2^2
\end{aligned}$$

The max of $-||Y - \mathbb{X}\,\beta||_2^2$ is the same as the min of $||Y - \mathbb{X}\,\beta||_2^2$

But this is just least squares!

Penalized maximum likelihood

Therefore, when we write

$$\hat{\beta}_{lasso}(\lambda) = \operatorname*{argmin}_{\beta} ||\mathbb{Y} - \mathbb{X}\beta||_2^2 + \lambda||\beta||_1$$

we can really think about it as

$$\hat{\beta}_{lasso}(\lambda) = \operatorname*{argmin}_{\beta} -\ell(\beta) + \lambda||\beta||_1$$

This holds for any (log) likelihood ℓ that we choose

Penalized maximum likelihood with binomials

Again, suppose we have data $(X_1, Y_1), \ldots, (X_n, Y_n)$

But this time, let $Y_i \in \{0, 1\}$

Now, assume that $Y_i | X_i \sim \mathrm{Bernoulli}(\pi(X_i))$

We can fully specify the likelihood by writing:

$$\log\left(\frac{\pi(X_i)}{1 - \pi(X_i)}\right) = X_i^\top \beta$$

This specifies a likelihood for Y_i conditional on X_i that has parameter vector β.

Penalized maximum likelihood with binomials

Finding the $\max \ell(\beta)$ in this case is called logistic regression

But, we can just as easily write the lasso version, with this new likelihood

$$\hat{\beta}_{lasso}(\lambda) = \underset{\beta}{\text{argmin}} -\ell(\beta) + \lambda||\beta||_1$$

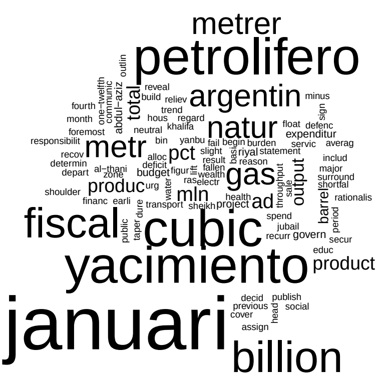
This allows us to leverage the nice properties of lasso while considering different types of responses

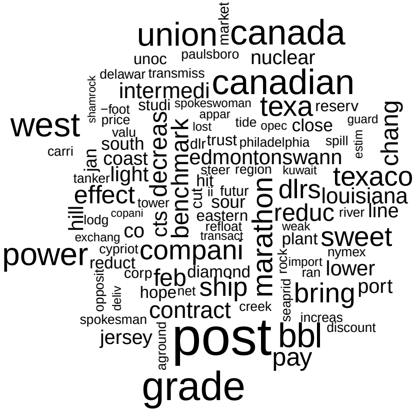
Penalized maximum likelihood with binomials: in R

```
require(glmnet)
Y = c(0,0,0,0,1,1,1,1)
out.glmnet = glmnet(y=Y,x=mydtm[1:8,],family='binomial')
out.cv.glmnet = cv.glmnet(y=Y,x=mydtm[1:8,],family='binomial')

beta.glmnet = out.glmnet$beta[,100]
words.glmnet = colnames(dtmTMNT)[abs(beta.glmnet)>0]

beta.glmnet.nonzero = beta.glmnet[abs(beta.glmnet)>0]

> round(beta.glmnet.nonzero,2)
    art  artist  christ   claim   mirag  patron   turtl
  46.05  154.91  202.12 1675.86  -32.39 7453.69  -13.70
```

Word clouds

Word clouds for LSI: Routers articles

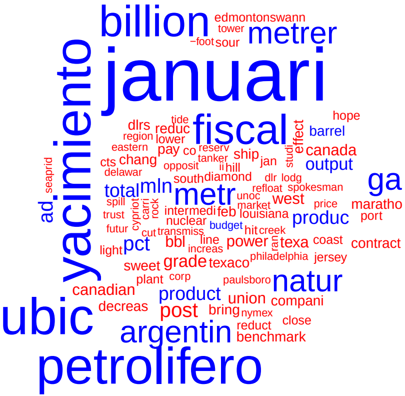


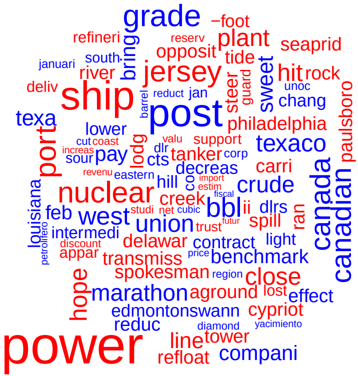


Largest values for term-concept, 1st column

Smallest values for term-concept, 1st column

Word clouds for LSI: Routers articles using color





abs(termConcept[,1]): positive, negative

abs(termConcept[,2]): positive, negative

Word clouds for TMNT: Logistic lasso

Plot of $\hat{\beta}$ from glmnet. positive, negative