Data Science for Business
Lecture #4
Topic Modeling for Movie Reviews Problem

Prof. Alan L. Montgomery

The University of Hong Kong & Carnegie Mellon University, Tepper School of Business

email: alanmontgomery@cmu.edu

Outline

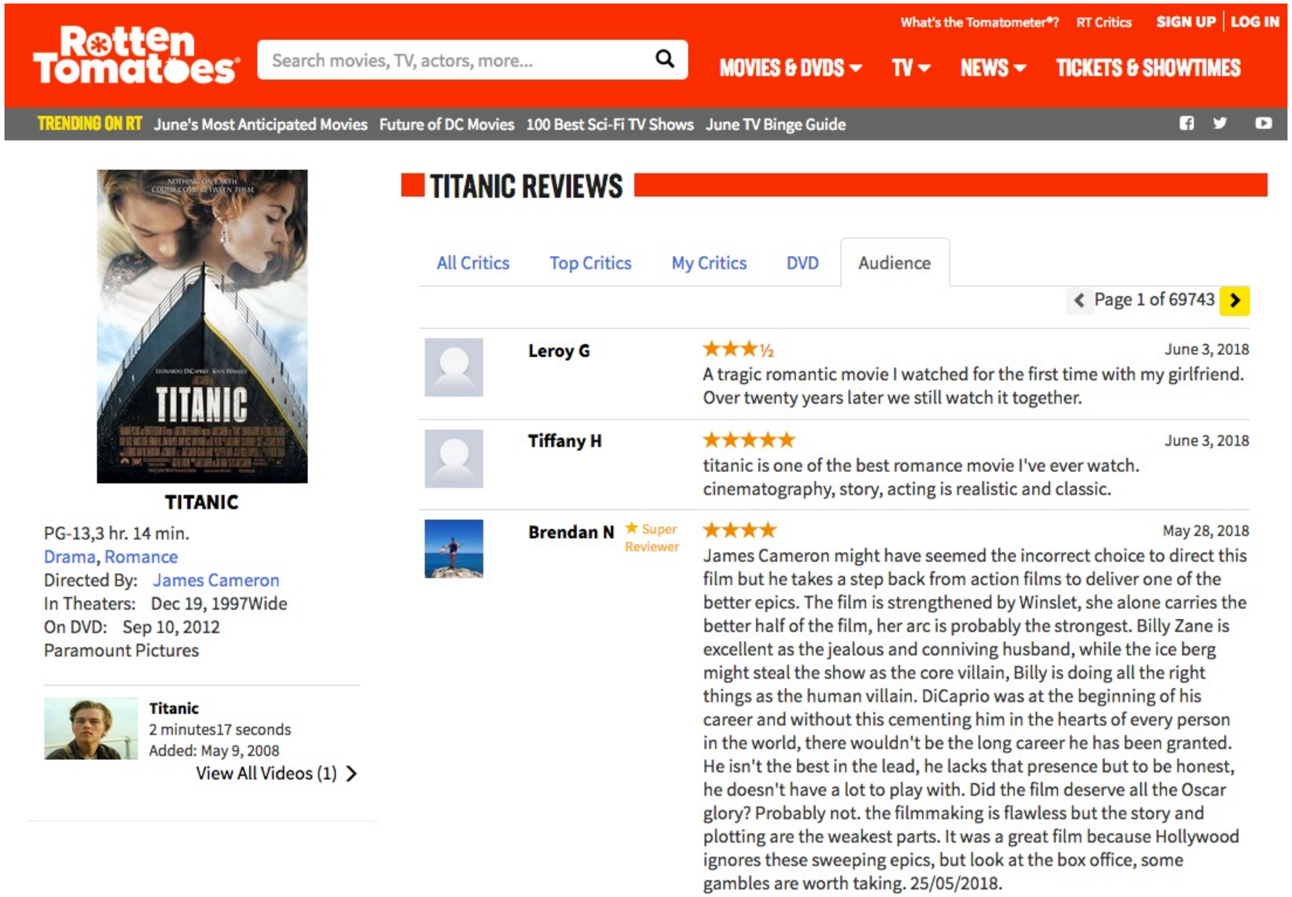
Clustering Movies using Text from Movie Reviews

Small Example for Topic Modeling applied to Movie Reviews

Clustering Movies using the Text from Movie Reviews

Need to structure “unstructured data/big data”

How do we analyze user reviews?



An approach for structuring is to map text to "tags"

Tags can be user generated like on Flickr or del.icio.us or tagged automatically

James Cameron might have seemed the incorrect choice to direct this film but he takes a step back from action films to deliver one of the better epics. The film is strengthened by Winslet, she alone carries the better half of the film, her arc is probably the strongest. Billy Zane is excellent as the jealous and conniving husband, while the ice berg might steal the show as the core villain, Billy is doing all the right things as the human villain. DiCaprio was at the beginning of his career and without this cementing him in the hearts of every person in the world, there wouldn't be the long career he has been granted. He isn't the best in the lead, he lacks that presence but to be honest, he doesn't have a lot to play with. Did the film deserve all the Oscar glory? Probably not. the filmmaking is flawless but the story and plotting are the weakest parts. It was a great film because Hollywood ignores these sweeping epics, but look at the box office, some gambles are worth taking. 25/05/2018.

- James Cameron
- action
- Winslet
- Billy Zane
- iceberg
- villain
- DiCaprio
- Oscar
- flawless
- epic

User generated “Tags” can be used to understand movies

Example of Information from MovieLens about Titanic summed across users

	A	B	C
1	odid	tag	count
2	10100	romance	68
3	10100	leonardo dicaprio	46
4	10100	atmospheric	29
5	10100	disaster	27
6	10100	love story	27
7	10100	true story	25
8	10100	drama	23
9	10100	kate winslet	23
10	10100	bittersweet	22
11	10100	oscar (best picture)	22
12	10100	historical	21
13	10100	catastrophe	20
14	10100	chick flick	15
15	10100	based on a true story	14
16	10100	sentimental	14
17	10100	james cameron	12
18	10100	nudity (topless - notable)	11
19	10100	nudity (topless)	10
20	10100	oscar (best cinematography)	9
21	10100	shipwreck	9
22	10100	survival	8
23	10100	oscar (best directing)	6
24	10100	action	5
25	10100	history	5
26	10100	music	5
27	10100	overrated	5
28	10100	time travel	5
29	10100	love	4
30	10100	kathy bates	3
31	10100	70mm	2
32	10100	big budget	2
33	10100	class differences	2
34	10100	epic	2
35	10100	girlie movie	2
36	10100	natural disaster	2

These are our **Words**. Each document only contains a subset of all the words in the vocabulary (e.g., Titanic uses 47 out of 962 terms). But these words tend to group together (e.g., “romance” and “love” tend to go together).

This yields a *huge* matrix of 962 terms by 1,153 movies

```
> mat_dtm[1:15,toptags]
```

Docs \ Terms	sci-fi	action	funny	visually appealing	superhero	animation	comedy	based on a book	predictable	twist ending
Titanic	0	5	0	0	0	0	0	0	1	0
The Dark Knight	0	57	0	0	97	0	0	0	0	0
Star Wars Ep. I: The P	42	4	0	1	0	0	0	0	0	0
Pirates of the Caribbe (2006)	0	8	7	0	0	0	14	0	0	0
Transformers: Revenge	3	12	1	0	0	0	2	0	0	0
Jurassic Park	58	32	5	0	0	0	0	21	1	0
Finding Nemo	0	0	22	0	0	51	10	0	4	0
Spider-Man 3	0	5	0	0	18	0	0	0	0	0
The Lion King	0	0	0	0	0	44	0	0	0	0
Shrek the Third	0	0	0	0	0	2	2	0	0	0
Transformers	10	9	0	0	0	0	1	0	0	0
Iron Man	29	23	21	9	74	0	2	0	1	0
Indiana Jones and the	0	1	0	0	0	0	1	0	0	0
Pirates of the Caribbe (2007)	0	2	2	0	0	0	13	0	0	0
Harry Potter and the H	0	0	6	0	0	0	11	13	0	0

Our objective is to reduce the dimensionality of the matrix and find interesting patterns. In other words group the words/rows and also group the movies/columns. Could use SVD which is often used for recommender systems.

Data: Movie Tags

```
> as.matrix(mterms[1:15,topterms])
                     animation based on a book comedy funny nudity (topless) predictable remake
Titanic                      0               0      0     0               10           1      0
The Dark Knight              0               0      0     0                0           0      0
Star Wars Ep. I: The         0               0      0     0                0           0      0
Pirates of the Carib         0               0     14     7                0           0      0
Transformers: Reveng         0               0      2     1                0           0      0
Jurassic Park                0              21      0     5                0           1      0
Finding Nemo                51               0     10    22                0           4      0
Spider-Man 3                 0               0      0     0                0           0      0
The Lion King               44               0      0     0                0           0      0
Shrek the Third              2               0      2     0                0           0      0
Transformers                 0               0      1     0                0           0      0
Iron Man                     0               0      2    21                0           1      0
Indiana Jones and th         0               0      1     0                0           0      0
Pirates of the Carib         0               0     13     2                0           0      0
Harry Potter and the         0              13     11     6                0           0      0
```

Relative Frequency of Words

```
> round(mtxterms[1:15,topterms],2)
```

	animation	based on a book	comedy	funny	nudity (topless)	predictable	remake
Titanic	0.00	0.00	0.00	0.00	0.02	0.00	0
The Dark Knight	0.00	0.00	0.00	0.00	0.00	0.00	0
Star Wars Ep. I: The	0.00	0.00	0.00	0.00	0.00	0.00	0
Pirates of the Carib	0.00	0.00	0.06	0.03	0.00	0.00	0
Transformers: Reveng	0.00	0.00	0.01	0.01	0.00	0.00	0
Jurassic Park	0.00	0.03	0.00	0.01	0.00	0.00	0
Finding Nemo	0.15	0.00	0.03	0.06	0.00	0.01	0
Spider-Man 3	0.00	0.00	0.00	0.00	0.00	0.00	0
The Lion King	0.15	0.00	0.00	0.00	0.00	0.00	0
Shrek the Third	0.04	0.00	0.04	0.00	0.00	0.00	0
Transformers	0.00	0.00	0.00	0.00	0.00	0.00	0
Iron Man	0.00	0.00	0.00	0.04	0.00	0.00	0
Indiana Jones and th	0.00	0.00	0.01	0.00	0.00	0.00	0
Pirates of the Carib	0.00	0.00	0.06	0.01	0.00	0.00	0
Harry Potter and the	0.00	0.06	0.05	0.03	0.00	0.00	0

What problems can we address with our review data?

Movie recommendation system

- What is another movie that user hasn’t viewed that they might like?
- Look for similar movies

Planning first-run release schedule

- What is the best week to release a movie?
- Look for weeks in which other (announced) releases are dis-similar, in other words find less competition

Dimensionality reduction

- Create a new embedding space that compresses our high-dimensional tag vector into a low dimension numeric space that preserves as much information as possible

Clustering Movies with Topic Models

Topic models or Latent Dirichlet Allocation models are a type of probabilistic clustering algorithm well suited for sparse text data

Latent Dirichlet Allocation detects "topics" which provide latent dimensions to describe observations

Word selection for document *i* at position *t*

$$w_{it} \sim \mathrm{M}(\beta_{z_{it}})$$

Multinomial Choice across all words based upon selected topic, β defines probabilities across words

Topic selection document *i* at time *t*

$$z_{it} \sim \mathrm{M}(\theta_i)$$

Multinomial Choice across all topics based upon unique document profile. Notice documents may choose more than one topic.

Unique Topic score shrunk across document *i*

$$\boldsymbol{\theta}_i \sim \mathrm{Dir}(\boldsymbol{\eta})$$

Dirichlet gives some structure to relationships between topics. What statisticians refer to as "shrinkage".

Understanding our topic model with a simple example

Suppose we observe the following three movies

Movie	"Adventure"	"Funny"
Jurassic Park	50	0
Iron Man	10	10
Finding Nemo	5	25

This is our raw data that has the counts of the tags

What do our topics mean?

Our LDA model learns the probability of a tag given a topic:

Tag	Topic1	Topic2
Adventure	100%	10%
Funny	0	90%

Each column corresponds with β or Prob(Word|Topic)

We can use unique topic profile to compare movies

Also our LDA model learns the distribution of topics associated with each movie:

Movie	Topic1	Topic2
Jurassic Park	100%	0
Iron Man	50%	50%
Finding Nemo	0	100%

Each row corresponds with θ or Prob(Topic) for a movie

Predicting the word count by multiplying tag distribution for each topic by the topic distribution

Our LDA model learns the probability of a tag given a topic:

Tag	Topic1	Topic2
Adventure	100%	10%
Funny	0	90%

Each column corresponds with β or Prob(Word|Topic)

The unique distribution of topics to Iron Man:

Movie	Topic1	Topic2
Iron Man	50%	50%

Each row corresponds with θ or Prob(Topic) for a movie

To construct predictions of the words counts we can multiply the conditional probability of words given the tags by the movies topic profile by word count:

Movie	"Adventure"	"Funny"
Topic 1	100% x 50%	0% x 50%
Topic 2	10% x 50%	90% x 50%
Total (weight by word count)	= (0.50 + 0.05) x 20 = 0.55 x 20 = 11	= (0 + 0.45) x 20 = 0.45 x 20 = 9

Assumes we know there are 20 tags in total:
we predict 11 Adventure and 9 Funny

This process illustrates how we move from the parameters to make inferences about the data. When training this model we make use of Bayes rule to invert the process and infer the parameters given the data.

Topic Modeling applied to the Movie Data with a Small Scale Dataset

See "movie_example.zip"

Setup and Read in the Data

First we need to load in the libraries that are needed. Topic Model often requires large matrices to be represented, but instead of representing this as a "dense" matrix we only store the values that are set using a "sparse" matrix

```
#################### setup environment  #####################

# setup libraries
if (!require(lattice)) {install.packages("lattice"); library(lattice)}
if (!require(NLP)) {install.packages("NLP"); library(NLP)}
if (!require(topicmodels)) {install.packages("topicmodels"); library(topicmodels)}
if (!require(tm)) {install.packages("tm"); library(tm)}
if (!require(slam)) {install.packages("slam"); library(slam)}




#################### input the data  #####################

## read in the data

# in RStudio select Menu Bar --> Session --> Set Working Directory --> To Source File Directory
# or automatically set working directory to be that of the script
setwd(dirname(rstudioapi::getActiveDocumentContext()$path))  # only works in Rstudio scripts
# alternatively set the working directory manually
#setwd("~/Documents/class/marketing analytics/cases/movies")  # !! edit and uncomment this line, if needed !!

# read in movie datasets
movies=read.delim("opus_movies.txt",header=T)  # the Opus movie data
tags=read.delim("opus_movielens_tags.txt",header=T)  # just the tags from movielens


## make modifications to the dataset

# change data formats
tags$odid=as.factor(tags$odid)
```

Transform the Data

I'll only save movies produced by Paramount and that use the three terms "action", "comic book" and "animation".

Our mterms matrix is a sparse matrix. You can reference the rows and columns using the selection as usual, but the result is another sparse matrix. If you want to work with the matrix in the usual way cast it as follows:

```
> as.matrix(mterms[1:2,1:2])
        action comic book
220100      23         31
5580100      0          0
> umovies$display_name[umovies$odid %in% c(220100,5580100)]
[1] Iron Man Beowulf
```

```
## transform the terms into a structure that can be used for topic modeling

# use this definition of mterms for movielens tags
# put data in sparse matrix form using simple_triplet_matrix as needed by LDA
mterms=simple_triplet_matrix(i=as.integer(tags$odid),j=as.integer(tags$tag),v=tags$count,
                             dimnames=list(levels(tags$odid),levels(tags$tag)))
# let's create a list of a smaller list of movies produced by paramount
movielist=movies$odid[movies$production_company1=="Paramount Pictures"]
# let's create a short list of terms to save
shorttermlist=c("action","comic book","animation")
# keep only the subset of a few terms
mterms=mterms[rownames(mterms) %in% movielist,shorttermlist]
# also delete any movies that do not have any terms
mterms=mterms[apply(mterms,1,sum)>0,]
# let's update the list of movies and their names since some might have just been deleted
movielist=rownames(mterms)
movienames=as.character(movies$display_name[movies$odid %in% rownames(mterms)])
# let's print out our matrix
as.matrix(mterms)
# let's lookup the movie names
movies[movies$odid %in% rownames(mterms),c("odid","display_name")]
# compute totals for mterms
lmterms=apply(mterms,1,sum)   # compute the sum of each of the rows (# of terms per movie)
lwterms=apply(mterms,2,sum)   # compute the sum of each of the columns (# of times word used)

# prepare a subset of movies with just the movies in our list
umovies=movies[movies$odid %in% as.integer(movielist),]   # create a subset of the movies that have terms
```

Perform a k-Means for Comparison

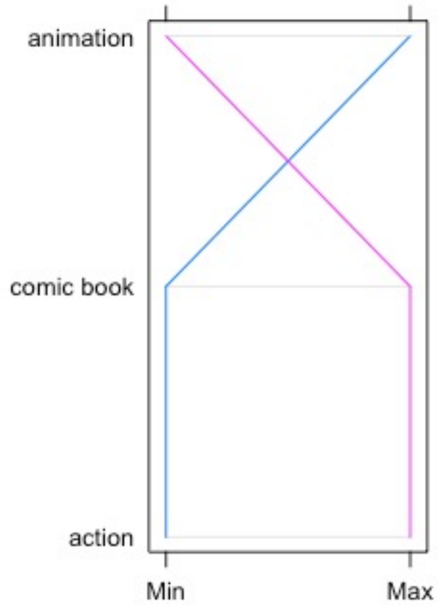
```r
#################### for reference compute kmeans cluster  #####################

# estimate kmeans with two topics
(grpKmeans=kmeans(mterms,centers=2))

# summarize the centroids
grpKcenter=t(grpKmeans$centers)
parallelplot(t(grpKcenter))

# print a table with the movies assigned to each cluster
for (i in 1:2) {
  print(paste("* * * Movies in Cluster #",i," * * *"))
  print(movienames[grpKmeans$cluster==i])
}
```

```
> results=cbind(as.matrix(mterms),grpKmeans$cluster)
> rownames(results)=umovies$display_name[rownames(results)%in%umovies$odid]
> results
                                  action comic book animation  
Iron Man                              23         31         0 2
Beowulf                                0          0         8 1
Transformers: Dark of the Moon         7          0         0 1
Puss in Boots                          0          0         6 1
Iron Man 2                            16         17         0 2
Captain America: The First Avenger    12         14         0 2
Rango                                  0          0        14 1
The Adventures of Tintin               8         12        21 1
True Grit                              2          0         0 1
G.I. Joe: Retaliation                  6          0         0 1
Hansel & Gretel: Witch Hunters         3          0         0 1
Jack Reacher                           9          0         0 1
```



Train our LDA Topic Model using LDA

```
#################### estimate an LDA topic model using keywords  #####################

## our first step is to estimate the topic model using LDA

# setup the parameters for LDA control vector
burnin=1000     # number of initial iterations to discard for Gibbs sampler (for slow processors use 500)
iter=5000       # number of iterations to use for estimation  (for slow processors use 1000)
thin=50         # only save every 50th iteration to save on storage
seed=list(203,5,63,101,765)  # random number generator seeds
nstart=5        # number of repeated random starts
best=TRUE       # only return the model with maximum posterior likelihood

# estimate a series of LDA models (each run can take a few minutes depending upon your processor)
ClusterOUT = LDA(mterms,2,method="Gibbs",control=list(nstart=nstart,seed=seed,best=best,burnin=burnin,iter=iter,thin=thin))
```

Caution: With a large dataset and a large number of topics this may take hours (or days). Test with small values first.

Output from LDA gives us Prob(Topic) for each movie and Prob(Word|Topic) for each topic

```
## now that we have saved the LDA results to our ClusterOUT object we want to
## extract the topic information and look at them

# probability of topic assignments (each movie has its own unique profile)
# rows are movies and columns are topics
ClustAssign = ClusterOUT@gamma   # this is a matrix with the row as the movie and column as the topic
rownames(ClustAssign)=movienames  # set the movie titles as the row names
dim(ClustAssign)  # check the dimension of the cluster (movies X topics)
head(ClustAssign,n=10)   # show the actual topic probabilities associated with the first 10 movies

# matrix with probabilities of each term per topic
ClustTopics = exp(ClusterOUT@beta)      # notice that we use "@" to access elements in the object and not "$" since this is an S4 object
colnames(ClustTopics)=colnames(mterms) # the columns are the terms
dim(ClustTopics)                       # check dimensions of the topics
print(ClustTopics)                     # print out clusters (topics in rows and terms in columns)
```

Movies are Mixtures of Topics

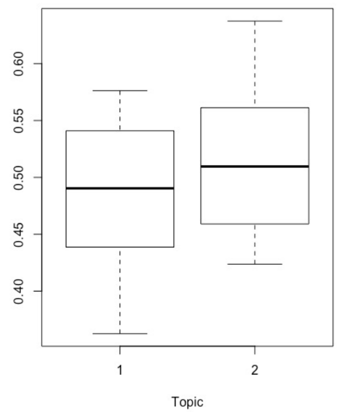
```r
## let's work on understanding the cluster based upon the movies

# visualize the distribution of topics across the movies
boxplot(ClustAssign,xlab="Topic",ylab="Probability of Topic across Movies")

# print a table with the movies assigned to each cluster
ClustBest = apply(ClustAssign,1,which.max)  # determine the best guess of a cluster, a vector with best guess
for (i in 1:2) {
  print(paste("* * * Movies in Cluster #",i," * * *"))
  print(movienames[ClustBest==i])
}
```

```
> ClustAssign
                                     [,1]      [,2]
Iron Man                        0.4615385 0.5384615
Beowulf                         0.4310345 0.5689655
Transformers: Dark of the Moon  0.5614035 0.4385965
Puss in Boots                   0.4464286 0.5535714
Iron Man 2                      0.4939759 0.5060241
Captain America: The First Avenger 0.4868421 0.5131579
Rango                           0.3906250 0.6093750
The Adventures of Tintin        0.3626374 0.6373626
True Grit                       0.5192308 0.4807692
G.I. Joe: Retaliation           0.5535714 0.4464286
Hansel & Gretel: Witch Hunters  0.5283019 0.4716981
Jack Reacher                    0.5762712 0.4237288
```

```
[1] "* * * Movies in Cluster # 1  * * *"
[1] "Transformers: Dark of the Moon" "True Grit"
[3] "G.I. Joe: Retaliation"          "Hansel & Gretel: Witch Hunters"
[5] "Jack Reacher"
[1] "* * * Movies in Cluster # 2  * * *"
[1] "Iron Man"
[2] "Beowulf"
[3] "Puss in Boots"
[4] "Iron Man 2"
[5] "Captain America: The First Avenger"
[6] "Rango"
[7] "The Adventures of Tintin"
```

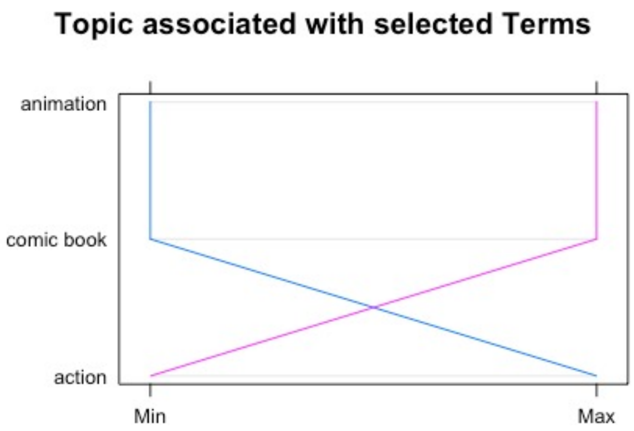


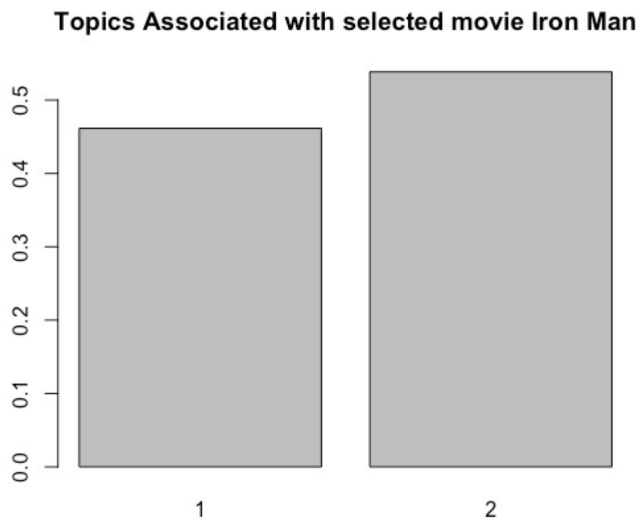
To understand a Topic look at its Associations

```r
## another way to understand the topics is through their associations with the keywords

# show the terms and associated topics
parallelplot(ClustTopics,main="Topic associated with selected Terms")

# show the topics associated with a selected movie
imovie=1
barplot(ClustAssign[imovie,],names.arg=1:ncol(ClustAssign),main=paste("Topics Associated with selected movie",umovies$display_name[imovie]))
```





We can predict the Words using our Model

```r
## last we can use our model to compute a best guess

# determine the best guess for each movie/term combination
ClustGuess=(ClustAssign%*%ClustTopics)*lmterms

# we can compare the predictions for a selected movie
imovie=1
mcompare=cbind(ClustGuess[imovie,],as.vector(mterms[imovie,]))
print(mcompare)

# or we can print the predictions for all movies
as.matrix(cbind(ClustGuess,mterms))

# compare kmeans solutions with the topic model
# remember that kmeans assignments are deterministic, while topic models are probabilistic
# so this cross tab only considers the matches between the most likely
xtabs(~grpKmeans$cluster+ClustBest)
```

```
> # or we can print the predictions for all movies
> as.matrix(cbind(ClustGuess,mterms))
                                    action comic book  animation action comic book animation
Iron Man                         24.888900 17.5033321 11.6077678     23         31         0
Beowulf                           3.443976  2.7394601  1.8165639      0          0         8
Transformers: Dark of the Moon    3.923207  1.8496469  1.2271458      7          0         0
Puss in Boots                     2.675058  1.9991935  1.3257489      0          0         6
Iron Man 2                       16.276970 10.0544180  6.6686120     16         17         0
Captain America: The First Avenger 12.639381  8.0329155  5.3277036     12         14         0
Rango                             5.462995  5.1333898  3.4036149      0          0        14
The Adventures of Tintin         14.854869 15.7217818 10.4233495      8         12        21
True Grit                         1.036835  0.5790622  0.3841031      2          0         0
G.I. Joe: Retaliation             3.315904  1.6135984  1.0704980      6          0         0
Hansel & Gretel: Witch Hunters    1.582380  0.8522703  0.5653493      3          0         0
Jack Reacher                      5.177514  2.2978569  1.5246291      9          0         0
```

Conclusion

Summary

Instead of clustering movies based upon attributes (like genre, movie stars, budget, ratings, …) we will cluster them using the text of movie reviews

We take the text of the movie reviews and pre-process them to yield “tags” that describe each movie. In our example, we have almost 1,000 terms that describe over 1,000 movies.

Topic modeling wishes to find “topics” or clusters that explain the kinds of words that we will see.

- The topics are common across movies, and you may find some topics are similar to genres like romance or science fiction, but a difference is that every word has some chance of showing up in a topic. Even “action” has a small chance of occurring in a romance.
- Every movie has a unique profile of topics, and this is what gives us our dimensionality reduction.

Unlike k-Means analysis where each observation is in one cluster, in topic modeling each observation has a chance of being assigned to every topic.