Text Mining and Data Analysis

Definition

- Text mining foes by numerous names (not always with the same meaning
  - Text Data Mining
  - Natural Language Processing
  - Unstructured Data Analysis
  - Text analytics

- Text mining is the process of identifying novel information from a collection of texts (also known as a corpus).

More specifically text mining…

- Functions with respect to a corpus of text (i.e. documents, texts, tweets)
- Creates a dictionary or vocabulary to identify relevant terms.
- Accommodates a variety of metrics to quantify the corpus

Data Mining vs. Text Mining

- Data Mining
  - process directly
  - Structured numeric transaction data residing in rational data warehouse
- Text Mining
  - Linguistic processing or natural language processing (NLP)
  - Discover heretofore unknown information
  - Applications deal with much more diverse and eclectic collections of systems and formats

Confusion

- Is text mining the same as information extraction?
  - No, but often they are used in unison.
- Information Extraction (IE)
  - Extract facts about pre-specified entities, events or relationships from unrestricted text sources.
  - Used to identify entities such as names, places, dates, zip codes.

The idea of text mining ...

- ... is to go beyond frequency-counting
- ... is to go beyond the search-for-documents framework
- The goal is to find patterns (of meaning) *within* and especially *across* documents.
  - Sometimes, even within sentances.

The steps of text mining

1. Data generation and application understanding
2. Corpus generation
3. Data understanding
4. Text preprocessing
5. Search for patterns / modelling
    - Topical analysis
    - Sentiment analysis / opinion mining
6. Evaluation

Corpus Development

- A corpus is a structured vectorizer form of textual data.
  - Tweets
  - Grant proposals
  - Books
    - Chapters within books
    - Sentences within paragraphs

Example 1: Getting Twitter Data

- Most social media sites allow the public access to a random sample of postings for analysis.
  - This includes Twitter, Instagram, and many others.
  - Doesn't include facebook anymore.

- In order to access Twitter data you must first go to dev.twitter.com and register.
  - It is free and only takes a few minutes

‘twitteR’ Package in R

- Provides a easy application program interface (API) to interface with twitter data.
  - You must set search terms (can set multiple using Boolean logic)
  - Can set date restrictions
  - Can set geographic restrictions using geocoding
  - Also can restrict language (*kinda..*)

```
tweets <- searchTwitter("energy + sustainability" , n=500, lang="en", since="2017-03-20")
```

Caveat 1: Sampling

- Only a small random sample of tweets are available.
- This means that if you try to get information on a specific locality (e.g. Tallahassee), there may not be any in the database.

```
tally<-searchTwitter('fsu', geocode='30.4419,84.2985,50mi',  n=50, since="2017-03-20")
```

- This search returns nothing.

Caveat 2: Geocoding

- Locations for tweets are only recorded if the person posting has allowed GPS access on their device and to Twitter.
  - This was only around 1% back in 2012.
  - The percentage has grown steadily but is probably still under 5% of the total.
  - Many social media researchers impute locations.
    - This works very well at the regional level with over 80% accuracy.

Twitter Data and Metadata

- Text: The actual tweet
- Favorited, favoritedCount: How many followers favorited the tweet
- Created: Date of tweet
- Retweet, retweetCount: How many times was it share
- Longitude, latitude: If geocoded, these are provided
- There is some other columns, such as source.

Examining Twitter Data in R

- After inputting data, it is usually best to convert the object to a data frame in R.
- After converting, we can examine the data.

```
tweets.df <- twListToDF(tweets3) #converts to dataframe
summary(tweets.df) # summarizes metadata
tweets.df[1:5,1] #prints first 5 tweets
```

Cleaning and Converting

- In order to use the text data, we must turn it into a corpula (collection of documents) and clean it.
- Cleaning involves:
  - Removing special characters
  - Removing punctuation
  - Standardizing case
  - Removing stopwords
  - Stemming

Removing “Specials”

- In social media analysis, we must also worry about emojis and foreign language text that is placed into twitter.
  - These can be frustrating and there is no comprehensive list of emojis.
- There are numerous methods, but I am going to make all text into ASCII. Thus removing the problem text.
  - This may not always be the best method.

R code

- It is easiest to remove special characters prior to creating the Corpus in my opinion.

```
tweets.df$text <- sapply(tweets.df$text,function(row)
iconv(row, "latin1", "ASCII", sub="")) #removes
emojis and foreign words


# Create a Corpus (text data set)
tweet_corpus <- Corpus(VectorSource( tweets.df[,1]
))
tweet_corpus #displays info on Corpus
```

Removing Punctuation and Capitalization

- Pretty simple.
- Please note, each time I do a cleanse I overwrite the original dataset. This is dangerous.

```
tweet_corpus <- tm_map(tweet_corpus, removePunctuation) # removes punctuation


tweet_corpus <- tm_map(tweet_corpus, content_transformer(tolower)) #makes all lower case
```

Stopwords

- The removal of words that add little to no value in data analysis.
- Most programs come with a preset list of common stopwords.
  - A, the, very, that, which
- Also includes words defined by the user.
  - For instance, you need to exclude your search terms. They will obviously be in every tweet

Zipf’s Law

- Zipf's law states that given some corpus of natural language utterances, the frequency of any word is inversely proportional to its rank in the frequency table.
  - Thus the most frequent word will occur approximately twice as often as the second most frequent word, three times as often as the third most frequent word, etc.:
  - Doesn’t apply to all languages, but many

Stemming

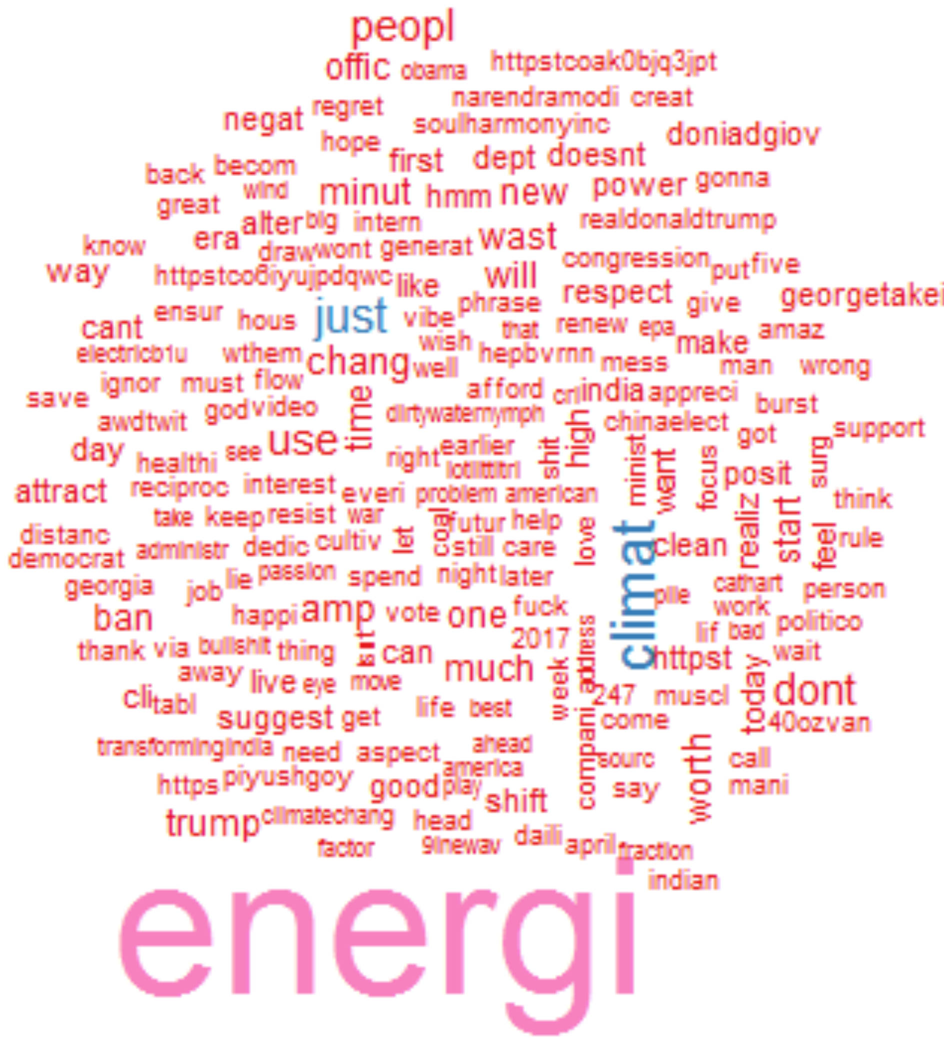
- Stemming removes the ends of words so that that conjugated (e.g. “ing”) and pluralized words end up having the same core.
  - We usually want the computer to think that “sustainability” and “sustainable” are the same.
- This process can also lead to some mistakes.
  - Will make “Miner” and “Mine” the same sometimes.

R Stemming and Stopword Code

```
tweet_corpus <- tm_map(tweet_corpus, function(x)removeWords(x,stopwords()))
#removes stop words

tweet_corpus <- tm_map(tweet_corpus, stemDocument)
```

Examining the Data in a Word Cloud

- Use a word cloud to examine the data after cleaning.
- It is usually best to exclude words that appear very few times (less than 2 at least)



energi
climat
just
peopl
offic
obama
httpstcoak0bjq3jpt
narendramodi
creat
negat
regret
soulharmonyinc
doniadgiov
hope
first
dept
doesnt
back
becom
wind
minut
hmm
new
power
gonna
great
alter
big
intern
realdonaldtrump
know
era
draw
wont
generat
wast
way
httpstco6iyujpdqwc
like
will
congression
put
five
respect
georgetakei
cant
ensur
hous
vibe
phrase
give
electricb1u
wthem
wish
that
renew
epa
make
amaz
chang
well
hepbvrnn
mess
man
wrong
ignor
must
flow
afford
india
appreci
save
god
video
time
dirtywaternymph
burst
awdtwit
chinaelect
support
day
use
high
got
healthi
see
right
earlier
shit
minist
posit
surg
attract
reciproc
interest
everi
problem
american
want
focus
think
take
keep
resist
war
coal
futur
help
realiz
start
distanc
administr
dedic
cultiv
let
still
care
love
clean
feel
rule
democrat
georgia
lie
passion
spend
night
later
cathart
person
job
amp
vote
one
fuck
work
ban
happi
bad
politico
thank
via
bullshit
thing
isnt
can
2017
address
httpst
lif
wait
away
live
eye
move
much
week
dont
cli
tabl
suggest
get
life
best
247
muscl
today
come
40ozvan
transformingindia
need
aspect
ahead
compani
sourc
call
https
piyushgoy
good
america
play
worth
mani
shift
say
trump
climatechang
head
daili
april
factor
9newav
fraction
indian

Removing Search Terms

- Search terms and words that appear too often make analysis difficult/impossible.
  - We want to remove the word “energy” because it is simply too frequently occurring.

```
tweet_corpus <- tm_map(tweet_corpus, function(x)removeWords(x,"energy"))  #removes additional term "energy"


wordcloud(tweet_corpus,min.freq=2, max.words=100, random.order=T, colors=pal2)
```


renew
develop
clean
amp
sourc
amplifi
siemensoilandga
via
global
convert
technolog
air
job
world
impact
time
guardian
solar
qualiti
quest
hous
renewableenergi
build
examin
2017
cleanenergi
portfolio
transit
irena
here
electr
may
just
power
can
latest
nuclear
effici
countri
fuel
use
market
need
learn
fleet
help
week
advanc
provid
work
low
mine
water
way
coal
zfep
take
encourag
save
wind
wef
transport
trend
invest
make
brightest
solut
innov
system
climat
motor
new
econom
will
like
zfep2018
growth
creat
enter
case
practic
famili
strong
rethink
busi
bus
follow
now
futur
show
home
get
cnbcenergi
report
voltag
execut
project
defect
paulsmithloma
green

Other stuff you can do…

- You can define terms that stay together
  - climate change as one term
  - Make EPA = Environmental Protection Agency
- Remove URLs
- Remove white space
- Remove numbers
- Entity Identification – finding people, places,…
  - Lexica, heuristic rules, syntax parsing
  - Jigsaw is a great free tool for this

Preparing for Text Analytics

- Once the data is cleaned, we need to transform the data into a document term matrix (dtm) for analysis.
  - DTMs basically develops a numeric representation of the textual data and analyzes it with standard tools.
  - The standard approach treats documents as rows and terms (e.g. words) as columns.
  - This creates a very large, sparse matrix (lots of zeros)
- You can also use a term document matrix

Document Term Matrix Issues

- Since the DTM is very sparse, we usually can't simply analyze it "as is".

1. Reduce the dimensionality: eliminating any words that appear too frequently or too infrequently.
2. Weight terms: Calculate a numerical statistic that is intended to reflect how important a word is to a document in a corpus.

Reducing Dimensionality

- Text mining suffers from too much information, so we want to reduce it down to something manageable.
- Words that appear in all the texts are useless at discriminating between factors.
  - For example, the word energy appears in every tweet, so it adds no value to choosing between them.
    - Think of a regression variable that equals 1 for 98% of your cases and 0 otherwise. It would be unlikely to be predictive.
  - Text mining usually begins by removing these too frequently occurring words.
- Words that appear in too few texts add little value also.
  - Not frequent enough to compare between groups.
    - Think of a regression variable that equals 1 for 3% of your cases and 0 otherwise. It would be unlikely to be predictive.

Weighting Terms

- The default weight is based on term frequency (how many times a word appears).
- Term frequency-inverse document frequency (tf-idf) is the preferred option to emphasize words that discriminate between factors.
  - The tf-idf value increases proportionally to the number of times a word appears in the document, but is often offset by the frequency of the word in the corpus.

R code for TDM

```
tdm <- TermDocumentMatrix(tweet_corpus,
        control =     list(wordLengths = c(1, Inf),
        weighting = function(x)
        weightTfIdf(x, normalize =FALSE) ) )
```

- The control function determines the inputs. You can also do stemming and stopwords here, I just prefer to do them early

Descriptive Stats on Term Freq

- It is useful to look at the most frequent terms

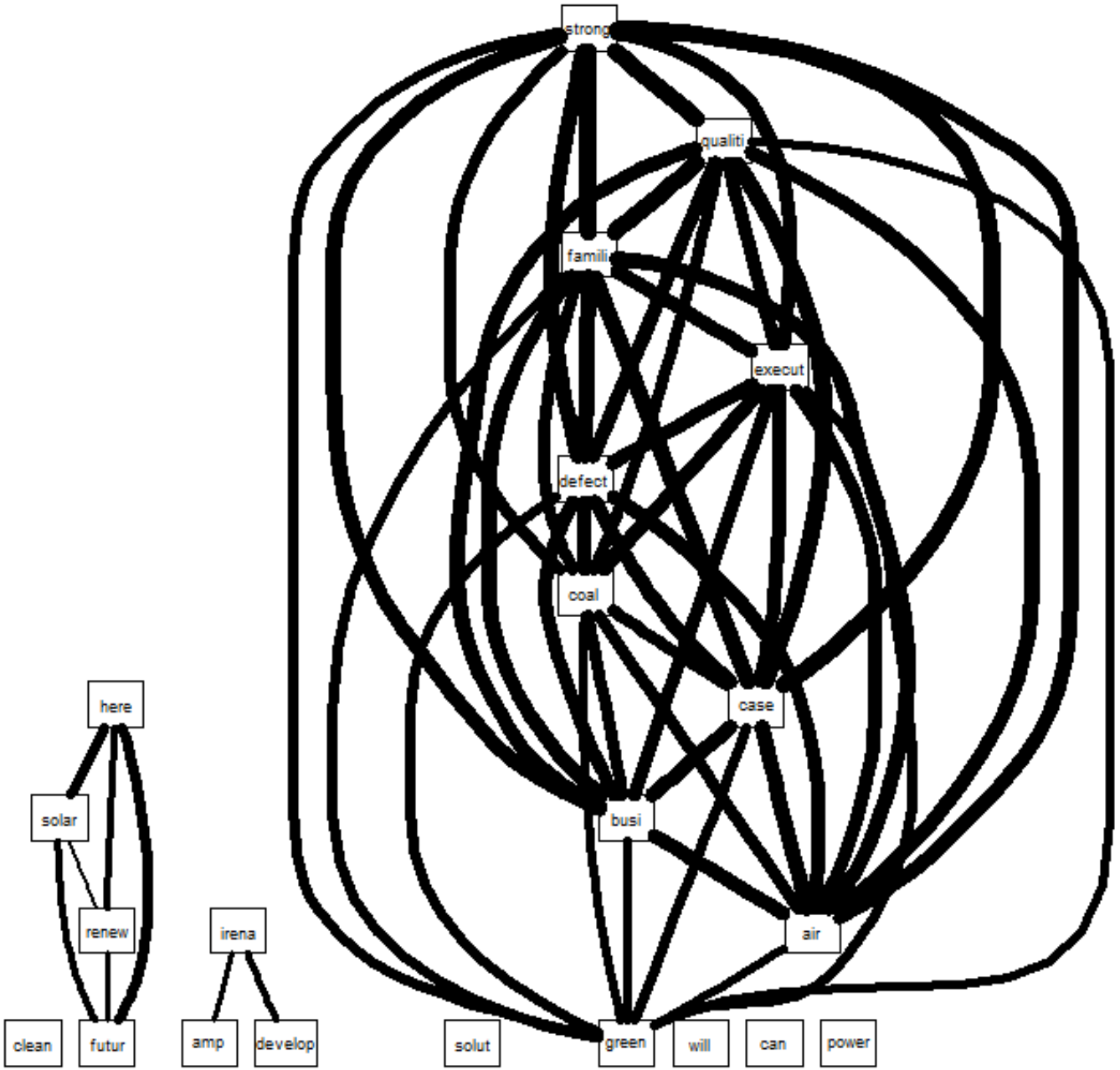
```
freq.terms <- findFreqTerms(tdm, lowfreq = 100)
term.freq <- rowSums(as.matrix(tdm))
term.freq <- subset(term.freq, term.freq >= 100)
df <- data.frame(term = names(term.freq), freq = term.freq)

ggplot(df, aes(x = term, y = freq)) + geom_bar(stat = "identity") + xlab("Terms") + ylab("Count") + coord_flip()
```

Examining Word Associations

- We may have a specific term we want to examine in relationship to others.
- I this case, I want terms that are over 20% correlated with “obama”

```
findAssocs(tdm, "obama", 0.2)
```

```
plot(tdm, term = freq.terms, corThreshold = 0.2,
weighting = T)
```


strong
qualiti
famili
execut
defect
coal
case
busi
air
here
solar
renew
clean
futur
irena
amp
develop
solut
green
will
can
power

Removing Sparse Terms

- You have two basic options here:
  1. Submit a list of low frequency words to exclude in the stopwords list
  2. Choose to dump those under a frequency or correlation threshold

```
tdm2 <- removeSparseTerms(tdm, sparse = 0.95)
m2 <- as.matrix(tdm2)
```

Ex 2: Getting Documents into a Corpus with Energy Plans

- R does not have a good text extraction tool for .pdf and .doc files.
  - The read.pdf() in the tm package assumes you have another program added on (i.e. xpdf engine)
  - Strongly suggest using another tool such as SAS enterprise miner or Jigsaw

Clustering

- Clustering is a method of partitioning data into ‘k’ subsets or groups.
  - Each observation is assigned to the closest cluster based on the distance to the center of the cluster.
  - Clustering is a form of unsupervised learning.
- Clustering indicates the group assignment, but the researcher must make sense of the meaning.

Clustering Terms

- After the data is cleaned, cluster analysis runs basically the same as numeric data.
  - You can not cluster effectively if the matrix is still sparse. Your results will be nonsense.
  - Clustering is based on a distance measure (e.g. Euclidian) same as in numeric data
  - You can use k-means, hierarchical, or Bayesian clustering.

Hieratical Cluster code

```
distMatrix <- dist(scale(m2))
fit <- hclust(distMatrix, method = "ward")
plot(fit)
rect.hclust(fit, k = 3)
m3 <- t(m2) # transpose the matrix to cluster documents (tweets)
set.seed(1222) # set a fixed random seed
k <- 3 # number of clusters
kmeansResult <- kmeans(m3, k)
```

K- Means Clustering Code

- kmeans3<-kmeans(dtm,3)

- #Merge cluster assignment back to keywords
- # kw_with_cluster <- as.data.frame(cbind(dtm, kmeans3$cluster))
- # names(kw_with_cluster) <- c("keyword", "kmeans3")

- #accumulator for cost results
- cost_df <- data.frame()

Elbow Plot

```
#run kmeans for all clusters up to 20
for(i in 1:20){
  #Run kmeans for each level of i, allowing up to 100 iterations for convergence
  kmeans<- kmeans(x=dtm, centers=i, iter.max=100)

  #Combine cluster number and cost together, write to df
  cost_df<- rbind(cost_df, cbind(i, kmeans$tot.withinss))
}
names(cost_df) <- c("cluster", "cost")

ggplot(data=cost_df, aes(x=cluster, y=cost, group=1)) +
theme_bw(base_family="Garamond") +
geom_line(colour = "darkgreen")
```

Determining Themes from Clusters

- You can extract key words for each cluster to determine the theme.
- This is very useful to check face validity.

```
for (i in 1:k) {
cat(paste("cluster ", i, ": ", sep = ""))
s <- sort(kmeansResult$centers[i, ], decreasing = T)
cat(names(s)[1:5], "nn")
}
```

Methods Notes

- Most software uses singular value decomposition (SVD)
  - SVD is basically uncentered principal components analysis.
  - So is it the same Aaron???
- The weighting scheme will impact clustering!
  - term frequency–inverse document frequency is the most robust in most of what I have read.
    - a numerical statistic that is intended to reflect how important a word is to a document in a corpus

Topic Analysis

- Topic modeling automatically classifying sets of documents into themes.
- Topic modeling is more specific than cluster analysis.
  - Cluster analysis can be used to model topics, but clusters do not necessarily map to topics.

Latent Dirichlet Allocation

(Blei .et al. 2003)

- LDA determines the latent topic structure given the words and document.
- LDA does this by recreating the documents in the corpus by adjusting the relative importance of topics in documents and words in topics iteratively.
- I prefer a Bayesian LDA model (Gibbs sampling) when possible for many reasons not gone through here.

Caveat to LDA and Cluster

- Both LDA and clustering require the researcher to set the number of topics/groupings.
- While there are methods to optimize this, the decision is always subjective.
- The number of clusters does NOT need to agree with the number of topics.

How LDA works

- Iteratively for each word(W), topic (T) and document (D) :
  - Randomly assigns words to topics.
  - Calculates a conditional probability of membership using:
    - Computes the P(T | D) – the proportion of words in D that are assigned to T
    - Computes the P(W| T) – the proportion of topic T over all documents contain word W
  - Reassign W based on p(T | D) * p(W | T)
  - Repeat thousands of times

What LDA Provides

- LDA considers each document to be a mixture of all topics.
- We get a probability for each topic and assume association with the max.
  - So you can just use the max to get a single classification or employ all probabilities in a more sophisticated model.

Document Classification

- Classification is different from clustering.
- Classification is when we have “experts” build a training data set by which other documents are judged.

- Classification is significantly more powerful Statistically and theoretically) than the unsupervised techniques we are learning.

Document Classification Steps

- Have human judge themes in text and classify.
  - More then one judge whenever possible.
- Develop a score for each document.
- Choose a classification method:
  - Rule builder, Neural network, Tree, Regression, Hierarchical Bayes
- Run all the data, including the training set, through the algorithm.
  - Check for how many are classified correctly.

Anomaly Detection in Text

- We may want to detect anomalies between a known document and others or over time.
- Over time:
  - Usually we look ate the use of a term or theme over time.
  - AnomalyDetection (R) is a useful tool combined with twitter feeds or any other time based textual analysis.

Anomalies comparison to other documents

- Usually, the classification techniques discussed earlier are used here.
  - Often combined with semantic analysis to see if opinion/mood changes (Called semantic similarity).
  - Rare event models can be applied if the anomalies are expected to occur infrequently.
  - Local Outlier Factors (LOF) are also useful in these settings.

Using Text Modeling in Research

- Text mining by itself is not very "publishable"
  - I highly recommend combining text with other data, particularly surveys.
- Do a sensitivity analysis!!!
  - Cluster analysis in particular is sensitive to starting points and other specifications.
  - I usually check to see if I get the same clusters by removing the top and bottom 5% and 10% for instance.

Clustering Research using Text Modeling

- What if my clustering methods don't agree?
  - This is the norm, not the exception.
  - Many ostensibly similar techniques are not actually asking the same question.
- Don't just look at what words are included in a cluster, look at what terms are excluded.
  - This often tells you more.

Text Mining in Regression

- The most frequent use of text mining is to get clusters and place them in a regression model.
  - Be careful if actually adjusting standard errors on these clusters because there will be misclassified units.
    - Outliers can really cause problems in some panel models.
  - Consider interacting the clusters with other variables.
    - Interactions often lead to me finding more misclassifications from the text analysis

Multiple Documents per Unit

- What if I have multiple documents per observation in my regression?
  - No perfect answer.
  - I suggest LDA in these cases because you get a score for each theme and can judge numerically which is closes.
  - If using clustering, you can use the mode.
  - If you want to be real advanced, create a multilevel model and keep both.

In vino veritas

Reference

- Wikipedia
- SAS: Text Mining Using SAS Text Miner
- https://cran.r-project.org/web/packages/tm/vignettes/tm.pdf
- Yanchang Zhao, Text Mining with R Twitter Data Analysis
- Ronen Feldman and Ido Dagan. “Knowledge discovery in textual databases (KDT)”, *Knowledge Discovery and Data Mining* (1995), 112-117.
- M. Hearst. “Untangling text data mining”. *Proceedings of ACL’99: the 37th Annual Meeting of the Association for Computational Linguistics*, 1999.
- Chen 2002
- Hsinchun Chen, “Knowledge Management Systems: A Text Mining Perspective”, *Knowledge Computing Corporation*, 2001.
- Douglass R. Cutting, Jan O. Pedersen, David Karger, and John W. Tukey. “Scatter/Gather: A cluster-based approach to browsing large document collections”. *In Proceedings of the 15th Annual International ACM/SIGIR Conference1992*, pages 318-329, Copenhagen, Denmark.
- Teuvo Kohonen. “The Self-Organizing Map”. Proceedings of the IEEE (1990), 78, 9, 1464-1480.
- J. Xu and W. B. Croft.
  1996. Query expansion using local and global document analysis. In *SIGIR '96: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 4-11, Zurich.
- Ray R. Larson. Bibliometrics of the world wide web: An exploratory analysis of the intellectual structure of cyberspace.
  In *ASIS '96: Proceedings of the 1996 Annual ASIS Meeting*.
- https://eight2late.wordpress.com/2015/05/27/a-gentle-introduction-to-text-mining-using-r/
- Blei .et al. 2003, “Latent Dirichlet Allocation”, Journal of Machine Learning Research 3 (2003) 993-1022