Text Mining

Theory and Applications

Anurag Nagar

Topics

- Introduction
- What is Text Mining
- Features of Text
- Document Representation
- Vector Space Model
- Document Similarities
- Document Classification and Clustering
- Software Packages for Text Mining
- Case Study of News Sentiment Analysis

Background

- Text data is everywhere – books, news, articles, financial analysis, blogs, social networking, etc.
- According to estimates, 80% of world's data is in unstructured text format [13].
- Need methods to extract, summarize, and analyze useful information from this data.
- Text Mining seeks to automatically discover useful knowledge from the massive amount of data.
- Lots of research going on in area of text mining in industry and academics.

What is Text Mining?

- Use of computational techniques to extract high quality information from text.
- Extract and discover knowledge hidden in text ***automatically*** (S. Ananiadou).
- **KDD definition** - “discovery by computer of new, previously unknown information, by automatically extracting information from a usually large amount of different unstructured textual resources.”

What is Text Mining?

- Application areas:

  - finding important concepts / ideas

  - finding key named entities

  - discovering associations between terms

  - generating hypothesis

  - summarizing large amount of textual and factual data.

  - link analysis

  - information retrieval

  - financial news analysis for stock prediction

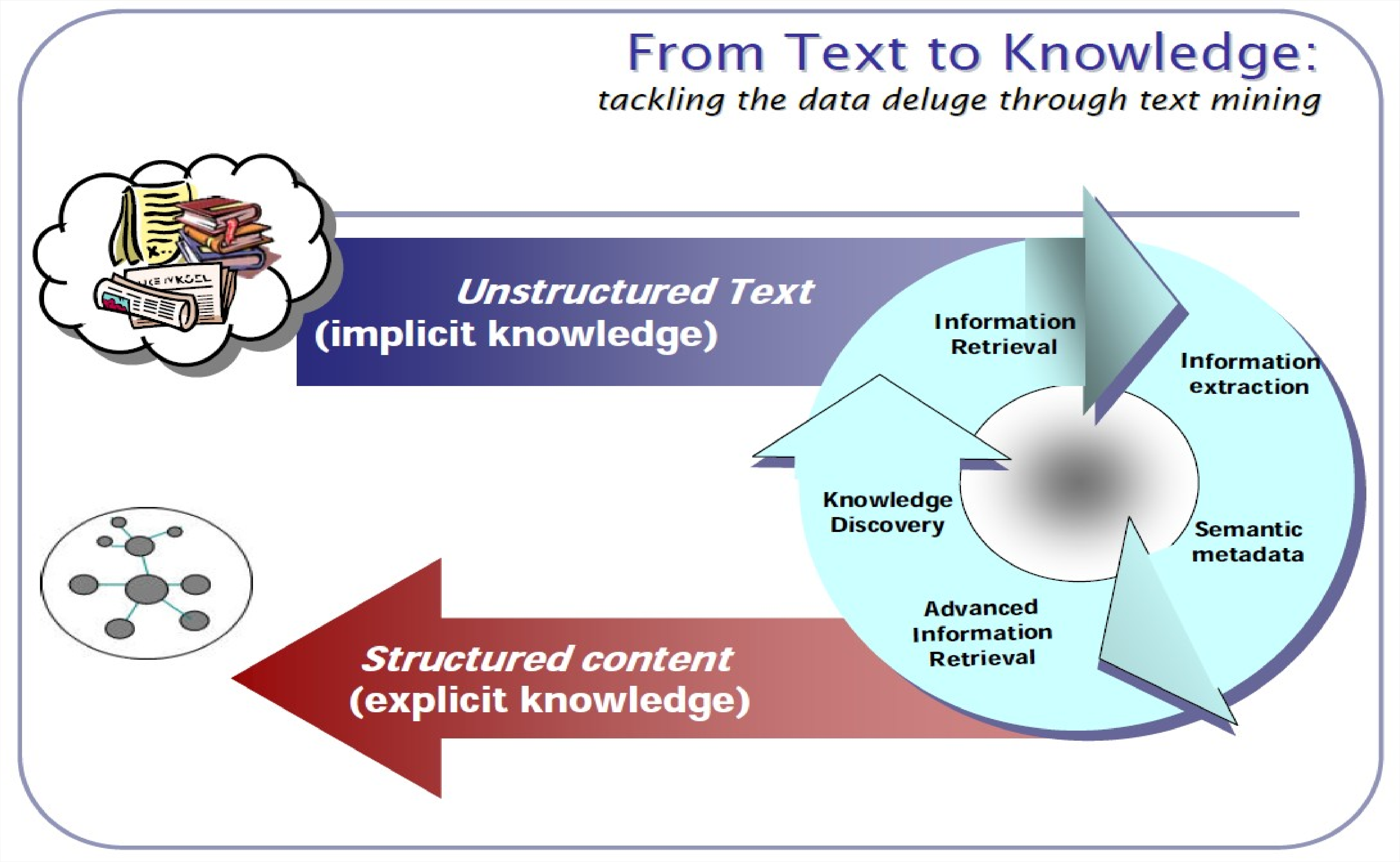
  - Text Categorization / Classification

  - Almost everything :)

Text Mining tasks

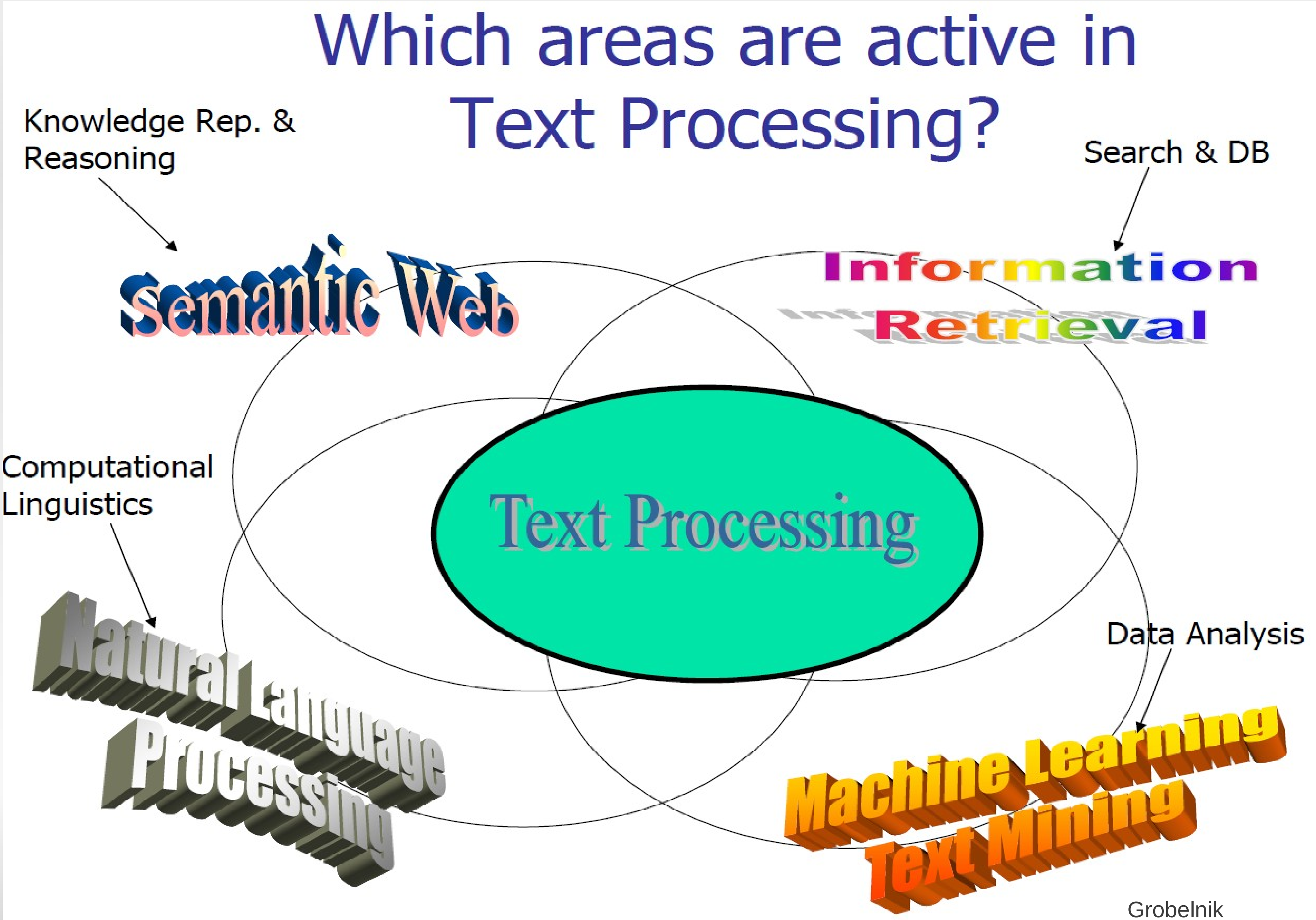
- Document Categorization *(supervised learning)*
- Document Clustering/Organization *(unsupervised learning)*
- Summarization *(key words, indices, etc.)*
- Visualization *(word cloud)*
- Numeric prediction *(stock market prediction based on news text)*

Text to Knowledge



S. Ananiadou

Text Mining Areas



Features of Text Data

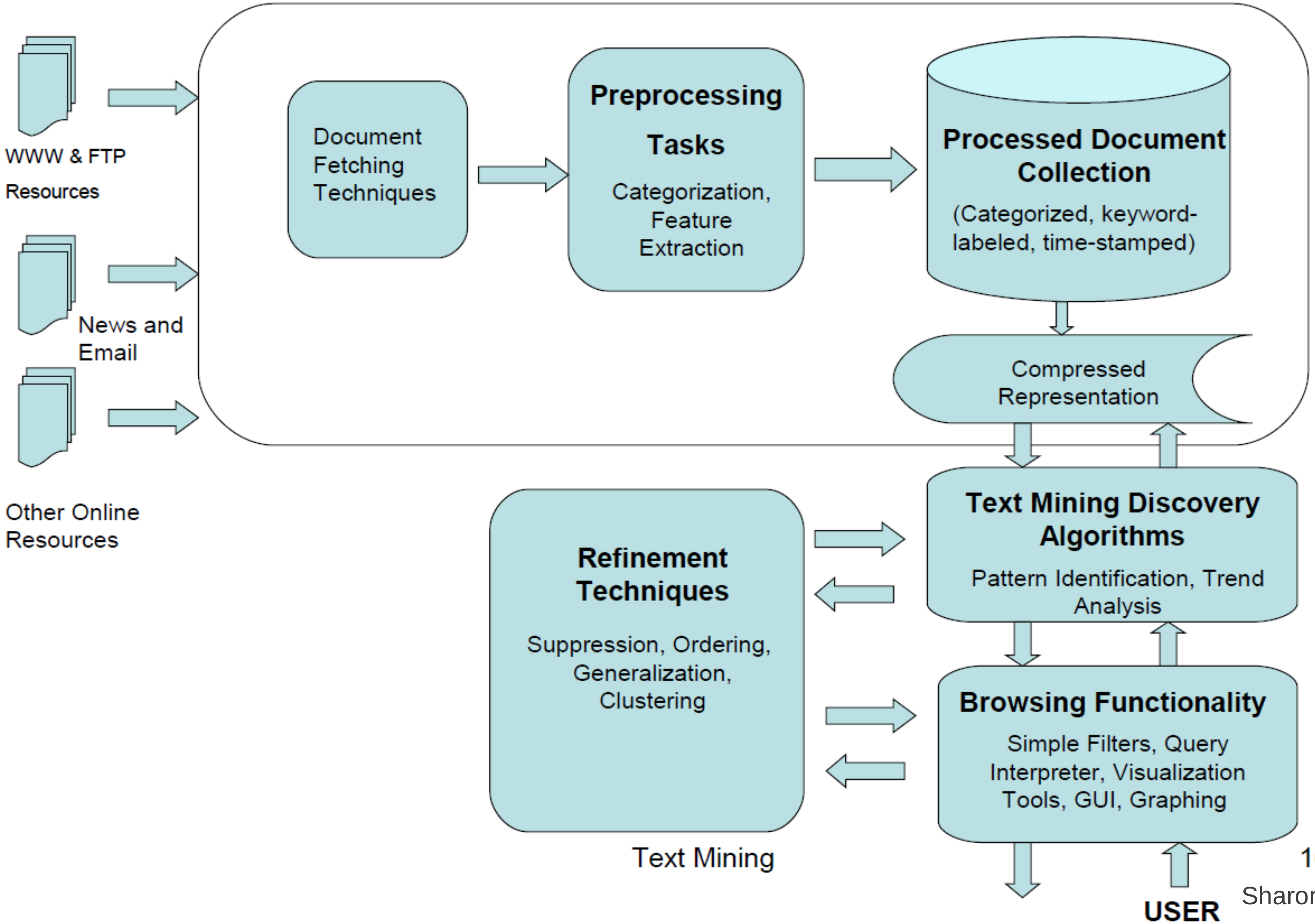
- High dimensionality
- Large number of features
- Multiple ways to represent the same concept.
- Highly redundant data.
- Unstructured data.
- Easy for humans, hard for machine.
- Abstract ideas hard to represent.
- Huge amount of data to be processed.

Text Properties

- Word properties:
  - Have relationships eg: synonyms, antonyms
  - Depend on context eg: the words “cold”, “run”
  - Have different forms eg: go, goes, went
  - Word frequencies in text have power distribution
   ...small number of very frequent words
   ...big number of low frequency words
  - Stop Words are often removed eg: a, the, to, what, etc

Architecture of TM Systems

The General Architecture of TMS:



Text Mining

Document Representation

- Any document can be represented by a list of terms and their associated weights:

  $D = \{(t_1,w_1),(t_2,w_2),...,(t_n,w_n)\}$

  where t_i is the i^{th} term and

  w_i is the weight for the i^{th} term

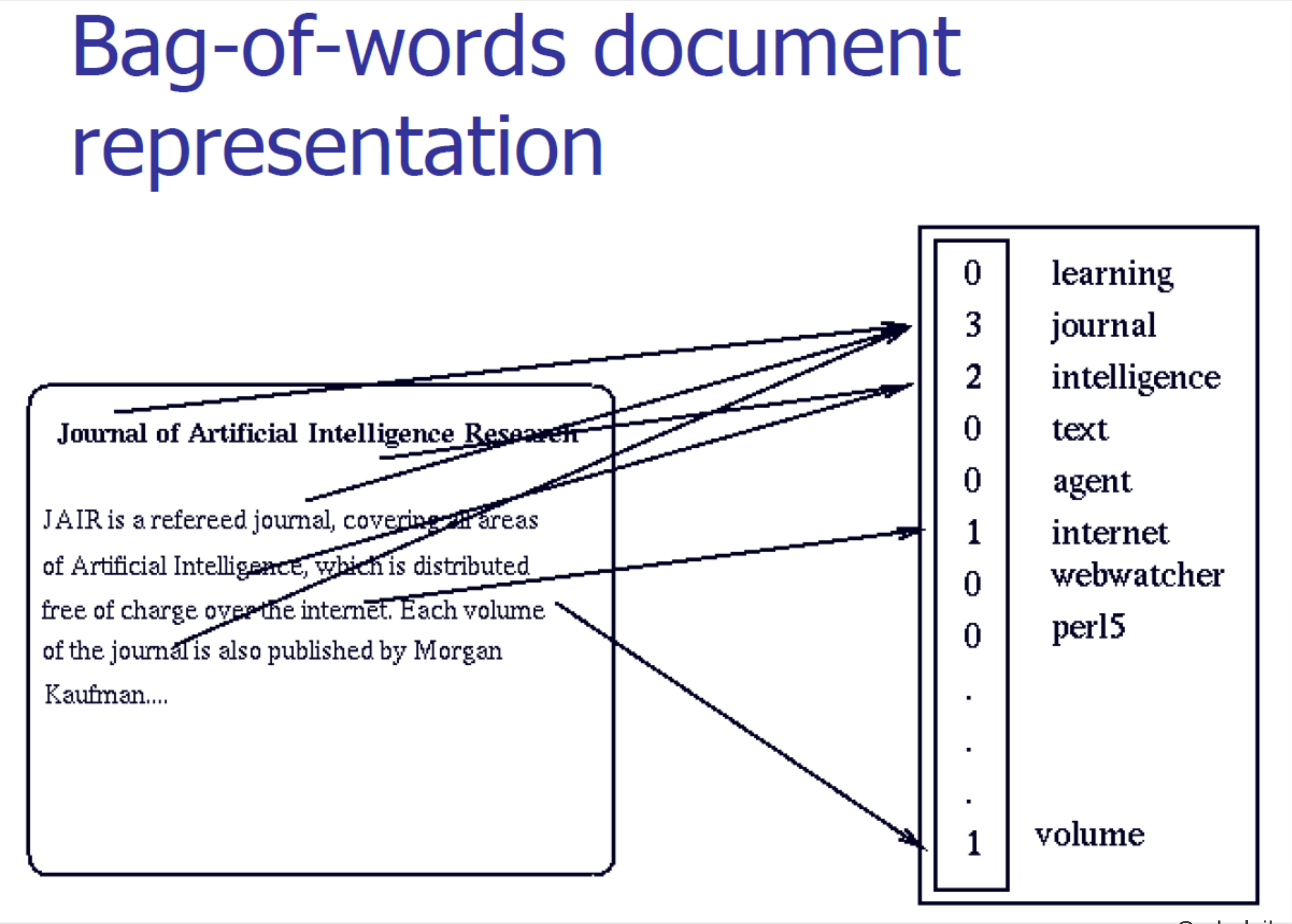
- Weight is a measure of the importance of a term in terms of information content.

Vector space model

- Documents are also treated as a “bag” of words or terms.
- Each document is represented as a vector.
- However, the term weights are no longer 0 or 1. Each term weight is computed based on some variations of TF or TF-IDF scheme.

- Term Frequency (TF) Scheme: The weight of a term t_i in document $\mathbf{d}_j$ is the number of times that t_i appears in $\mathbf{d}_j$, denoted by f_{ij}. Normalization may also be applied.

Bag of Words Representation



Grobelnik

TF-IDF term weighting scheme

- The most well known weighting scheme
  - TF: still **term frequency**
  - IDF: **inverse document frequency**.

  N: total number of docs

  df_i: the number of docs that t_i appears.
- The final TF-IDF term weight is:

$$tf_{ij} = \frac{f_{ij}}{\max\{f_{1j}, f_{2j}, \ldots, f_{|V|j}\}}$$

$$idf_i = \log \frac{N}{df_i}$$

$$w_{ij} = tf_{ij} \times idf_i.$$

TF-IDF Example

- Consider a document containing 100 words wherein the word cow appears 3 times. Following the previously defined formulas, the term frequency (TF) for cow is then (3 / 100) = 0.03. Now, assume we have 10 million documents and cow appears in one thousand of these. Then, the inverse document frequency is calculated as log(10 000 000 / 1 000) = 4. The tf*idf score is the product of these quantities: 0.03 × 4 = 0.12.

Wikipedia

TF-IDF Weighting Example

Example document and its vector representation

- TRUMP MAKES BID FOR CONTROL OF RESORTS Casino owner and real estate Donald Trump has offered to acquire all Class B common shares of Resorts International Inc, a spokesman for Trump said. The estate of late Resorts chairman James M. Crosby owns 340,783 of the 752,297 Class B shares. Resorts also has about 6,432,000 Class A common shares outstanding. Each Class B share has 100 times the voting power of a Class A share, giving the Class B stock about 93 pct of Resorts' voting power.
- [RESORTS:0.624] [CLASS:0.487] [TRUMP:0.367] [VOTING:0.171] [ESTATE:0.166] [POWER:0.134] [CROSBY:0.134] [CASINO:0.119] [DEVELOPER:0.118] [SHARES:0.117] [OWNER:0.102] [DONALD:0.097] [COMMON:0.093] [GIVING:0.081] [OWNS:0.080] [MAKES:0.078] [TIMES:0.075] [SHARE:0.072] [JAMES:0.070] [REAL:0.068] [CONTROL:0.065] [ACQUIRE:0.064] [OFFERED:0.063] [BID:0.063] [LATE:0.062] [OUTSTANDING:0.056] [SPOKESMAN:0.049] [CHAIRMAN:0.049] [INTERNATIONAL:0.041] [STOCK:0.035] [YORK:0.035] [PCT:0.022] [MARCH:0.011]

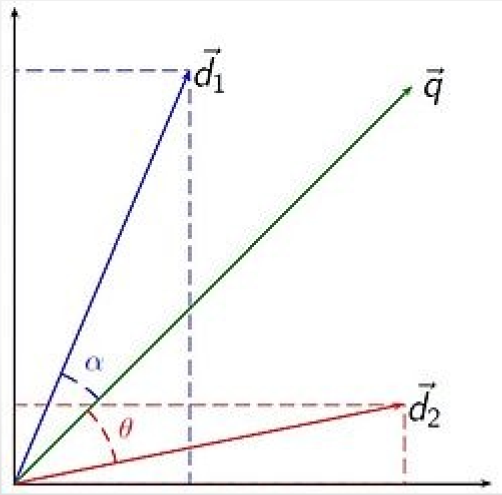
Grobelnik et al

Cosine Similarity

- Relevance of a query q (represented as a vector) to document d_i can be calculated using cosine similarity.
- Cosine Similarity can be computed as:

$$cosine(\mathbf{d}_j, \mathbf{q}) = \frac{\langle \mathbf{d}_j \bullet \mathbf{q} \rangle}{\| \mathbf{d}_j \| \times \| \mathbf{q} \|} = \frac{\sum_{i=1}^{|V|} w_{ij} \times w_{iq}}{\sqrt{\sum_{i=1}^{|V|} w_{ij}^2} \times \sqrt{\sum_{i=1}^{|V|} w_{iq}^2}}$$

It can be visualized as the cosine of the angle between the two documents when they are represented as a vector.



Word Frequencies

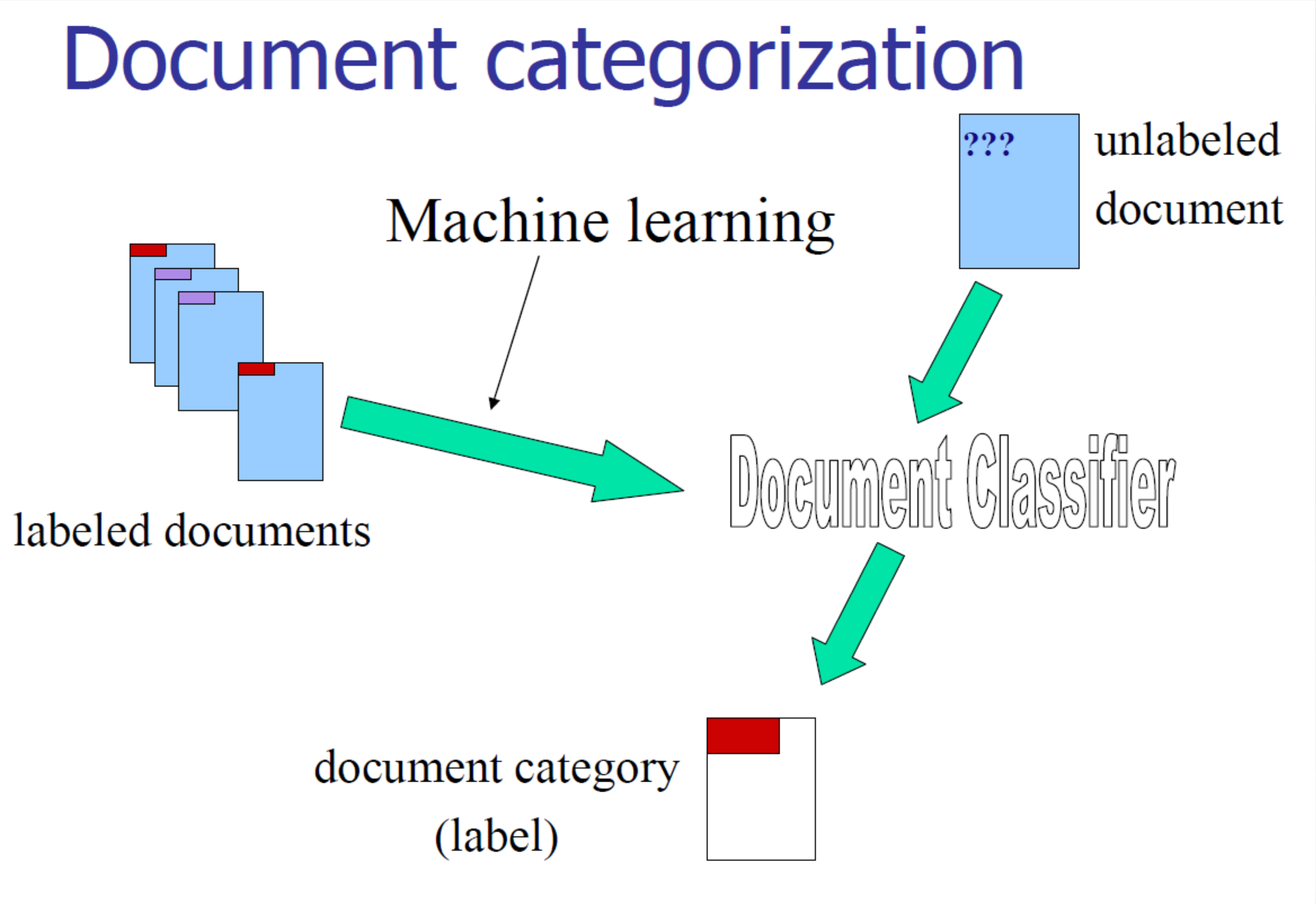
- **Zipf's Law:**
- Idea: We use a few words very often, and most words very rarely, because it's more effort to use a rare word.
- Zipf's Law: Product of frequency of word and its rank is [reasonably] constant.
- Empirically demonstrable. And holds up over different languages.

Zipf Law Example

Zipf's Law Example:

Word	*Rank*	*Freq.*	*Rank*F*	*Word*	*Rank*	*Freq.*	*Rank*F*
the	1	120021	120021	investors	400	828	331200
of	2	72225	144450	head	800	421	336800
and	4	53462	213848	warrant	1600	184	294400
for	8	25578	204624	Tehran	3200	73	233600
is	16	16739	267824	guarantee	6400	25	160000
company	32	9340	298880	Pittston	10000	11	110000
Co.	64	4005	256320	thinly	20000	3	60000
quarter	100	2677	267700	Morgenthaler	40000	1	40000
unit	200	1489	297800	tabulating	47075	1	47075

Classification / Categorization



Grobelnik et al

Classification Algorithms

1) k- Nearest Neighbors
2) Naive Bayes
3) Support Vector Machines

Traditional Algorithms

Newer Algorithms

4) Boos Texter – simple, relatively fast algorithm with excellent classification accuracy.
Based on an ensemble of classifiers approach.

Schapire, Robert E., and Yoram Singer. "BoosTexter: A Boosting-based System for Text Categorization." Machine Learning 39 (2000).

5) Based on Neural Networks, Decision Trees

K-Nearest Neighbor Classifier

K-Nearest Neighbor Classifier

Classification Problem:

Given: a training set of vectors $x_1, \ldots, x_n$ in R^m with their classifications (labels) $d_1, \ldots, d_n$ in $\{0,1\}$ (or in $\{0, \ldots, c\}$).

Problem: classify (assign labels to) vectors from a test set $y_1, y_2, y_3, \ldots$

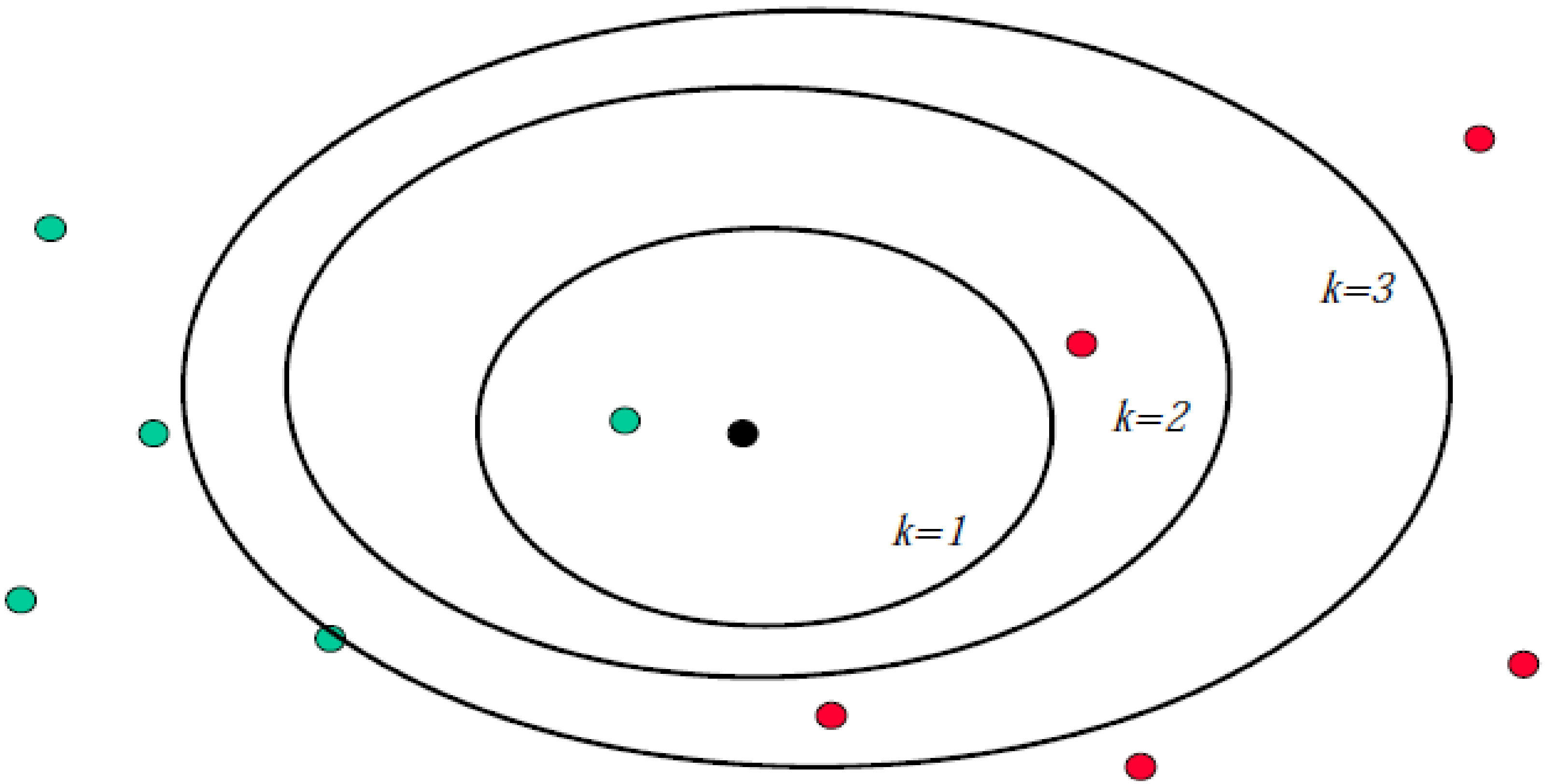
Main Idea:

"cases that are *close/similar* to each other should have the same labels"

to find a label for y find the distance between y and $\mathbf{x_1, x_2, \ldots, x_n}$; $\mathbf{x}$ which is closest to y (***nearest neighbor***) determines the label of y.

K-Nearest Neighbor Classifier

k-Nearest Neighbor Classifier



Instead of "the closest *x*" we look for "*k* closest *x*'s"; majority wins !15

Model Evaluation

- Precision is the fraction of retrieved instances that are relevant.
- Recall is the fraction of relevant instances that are retrieved

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

A measure that combines precision and recall is the F-score

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Software for Text Mining

Software for Text Mining

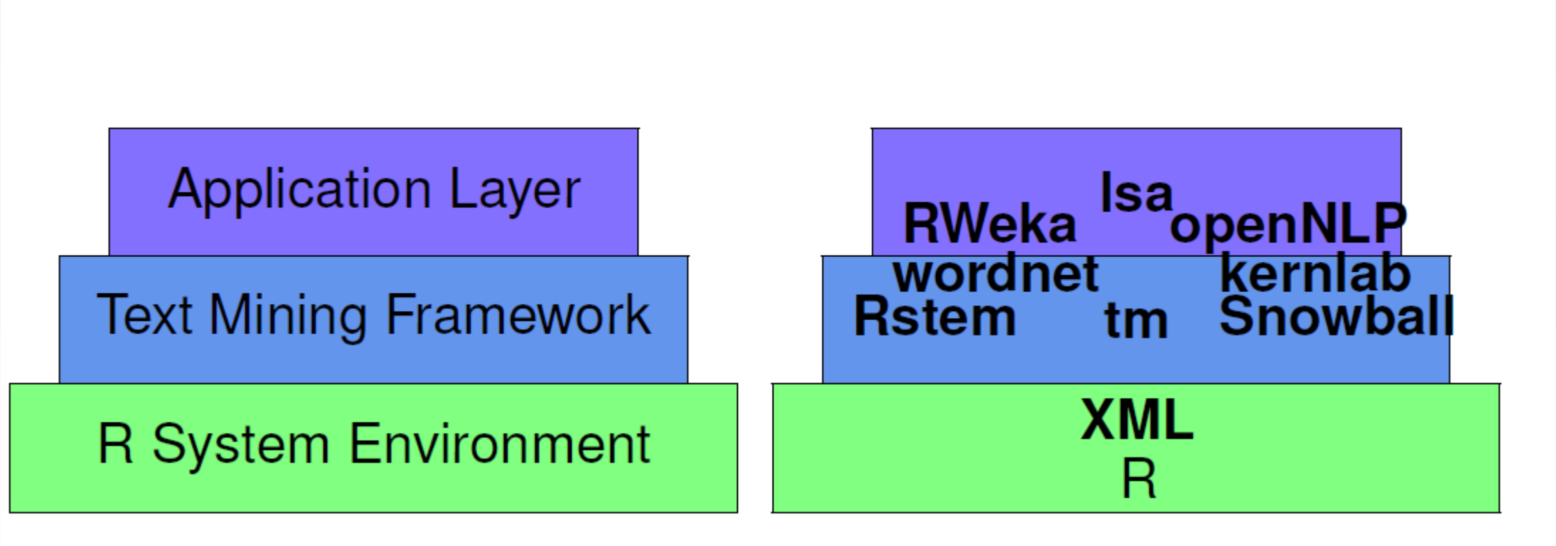
- A number of academic/commercial software available:
  1. SAS TextMiner
  2. IBM SPSS
  3. Number of open source packages in R – eg:tm
  4. Boos Texter
  5. StatSoft
  6. AeroText
- Text Data is everywhere – it needs to be mined !!

R package - tm

- “tm” package for text mining in R.
- Package offers functionality for managing text documents, performing analysis, and data mining.
- Numerous extensions and plugins have been developed for handling various formats of input data and producing summary results.

R package - tm

- Input is represented as a Corpus
- Various readers perform automated pre-processing, feature extraction, and load data into memory.



Conceptual Layers of R framework

Corpus Construction

1. Fetch documents from sources (disk, Internet)
2. Parse document structure (HTML, PDF, `getReaders()`)
3. Extract text and meta information
4. Dynamically create corpus
5. Fill corpus
   - immediately
   - delayed (load on demand)
   - referentially (using pointers to a database)

Package tm

- getReaders() gives available readers

an@vaio: ~/courses/AdvDM/TextMining

```
> getReaders()
 [1] "readDOC"                    "readGmane"
 [3] "readPDF"                    "readReut21578XML"
 [5] "readReut21578XMLasPlain" "readPlain"
 [7] "readRCV1"                   "readRCV1asPlain"
 [9] "readTabular"                "readXML"
>
```

- getSources() gives available sources

an@vaio: ~/courses/AdvDM/TextMining

```
> getSources()
[1] "DataframeSource" "DirSource"       "GmaneSource"     "ReutersSource"
[5] "URISource"       "VectorSource"
>
```

Loading data

Loading Data from text documents in a directory

```
> txt <- system.file("texts", "txt", package = "tm")
> (ovid <- Corpus(DirSource(txt),
+                  readerControl = list(language = "lat")))

A corpus with 5 text documents
```

```
>summary(ovid)
>inspect(ovid[1:2])
```

Loading data from the web

- Can use URISource to create corpus

```
> DosSource <- URISource("http://www.gutenberg.org/files/2554/2554.txt")
> Dostoevsky <- Corpus(DosSource)
> Dostoevsky[[1]][1]
[1] "The Project Gutenberg EBook of Crime and Punishment, by Fyodor Dostoevsky"
```

```
> tf <- termFreq(Dostoevsky[[1]])
> head(tf)
      --"that          --and           --he          --she --translator's     --zossimov
            1              2              1              1              1              1
```

Document Term Matrix

- Document-Term matrix contains documents as rows and terms as columns:

```
an@vaio: ~/courses/AdvDM/TextMining

> dtm<-DocumentTermMatrix(ovid)
> dtm
A document-term matrix (5 documents, 384 terms)

Non-/sparse entries: 419/1501
Sparsity           : 78%
Maximal term length: 22
Weighting          : term frequency (tf)
>
```

Document Term Matrix

- Can inspect the matrix created:

an@vaio: ~/courses/AdvDM/TextMining

```
> inspect(dtm[1:2,5:10])
A document-term matrix (2 documents, 6 terms)

Non-/sparse entries: 2/10
Sparsity           : 83%
Maximal term length: 8
Weighting          : term frequency (tf)

            Terms
Docs          adit: admissa aeacidae ae[illegible]ae aequore aëriae
  ovid_1.txt     0       0        0      0       0      0
  ovid_2.txt     0       0        1      0       0      1
>
```

Term Document Matrix (TDM)

- TDM contains terms as rows and documents as columns.
- Can also create chunks of terms and load.

an@vaio: ~/courses/AdvDM/TextMining

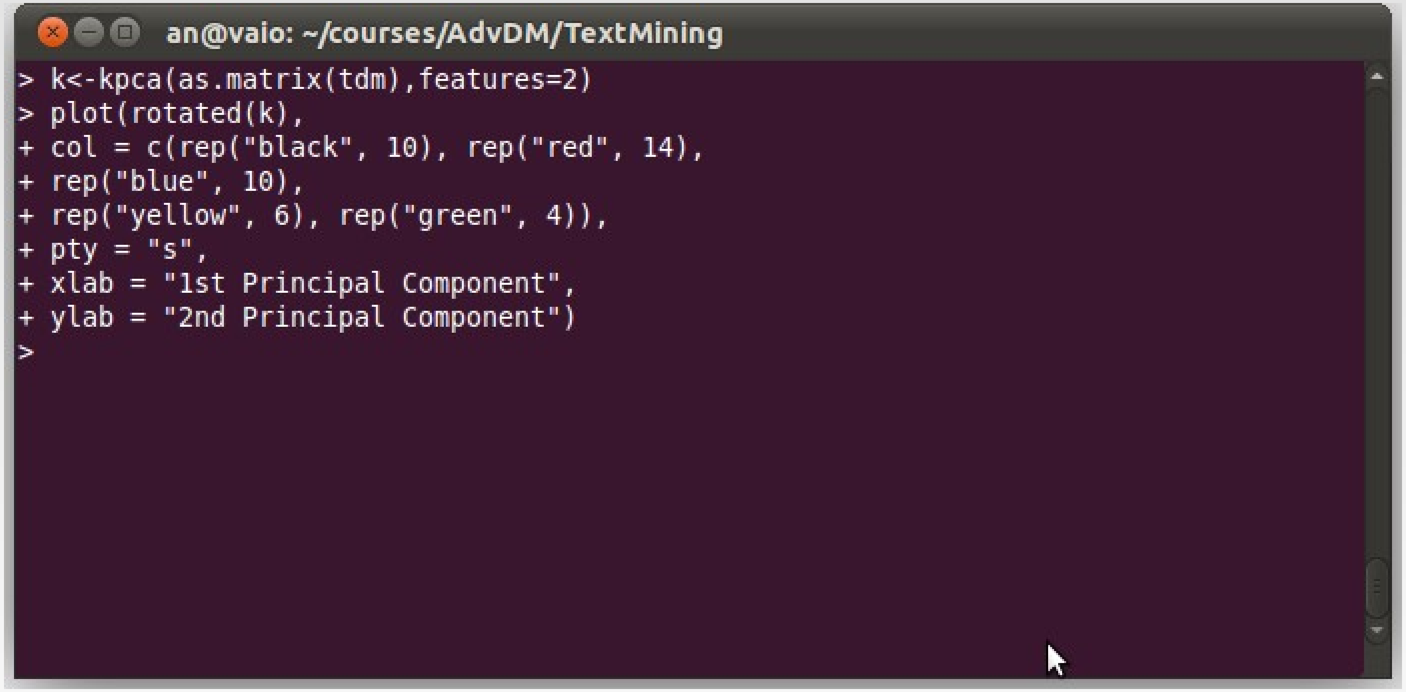
```
> tdm<-TermDocumentMatrix(makeChunks(ovid,500),list(weighting=weightBin))
> inspect(tdm[5:10,1:2])
A term-document matrix (6 terms, 2 documents)

Non-/sparse entries: 2/10
Sparsity           : 83%
Maximal term length: 8
Weighting          : binary (bin)

          Docs
Terms      1 2
  adit:    0 0
  admissa  0 0
  aeacidae 0 1
  aeneae   0 0
  aequore  0 0
  aëriae   0 1
> 
```

PCA Analysis using TDM

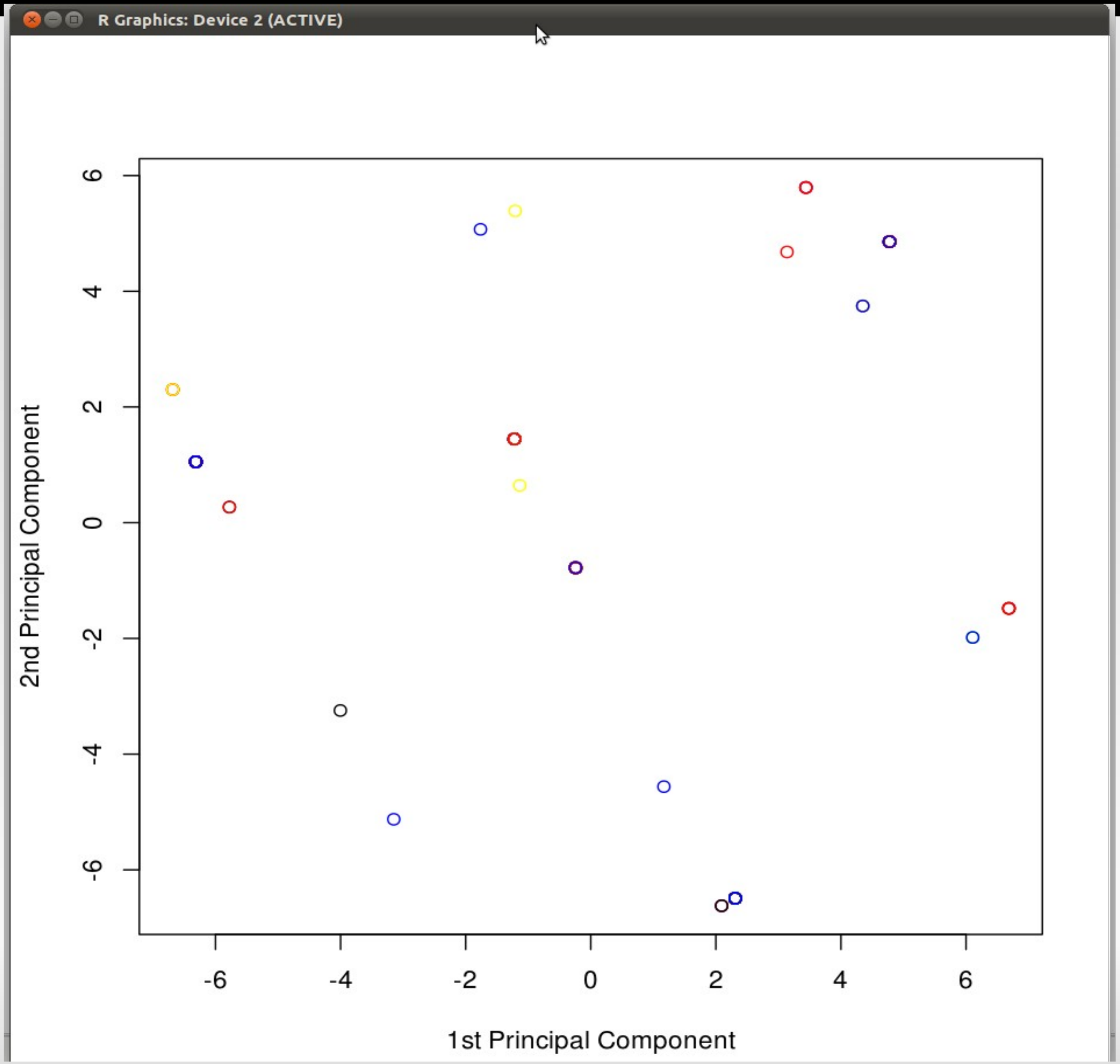
- TDM matrix can be used for PCA analysis and plotting



an@vaio: ~/courses/AdvDM/TextMining

```
> k<-kpca(as.matrix(tdm),features=2)
> plot(rotated(k),
+ col = c(rep("black", 10), rep("red", 14),
+ rep("blue", 10),
+ rep("yellow", 6), rep("green", 4)),
+ pty = "s",
+ xlab = "1st Principal Component",
+ ylab = "2nd Principal Component")
>
```

PCA plot using 2 features



Processing text

- Very easy to process and manipulate text.
- Stemming a document:

```
> sd <- stemDocument(Dostoevsky[[1]])
> head(sd)
[1] "The Project Gutenberg EBook of Crime and Punishment, by Fyodor Dostoevski"
[2] ""
[3] "This eBook is for the use of anyon anywher at no cost and with"
[4] "almost no restrict whatsoever.  You may copi it, give it away or"
[5] "re-us it under the term of the Project Gutenberg Licens includ"
[6] "with this eBook or onlin at www.gutenberg.org"
```

- Unlisting words:

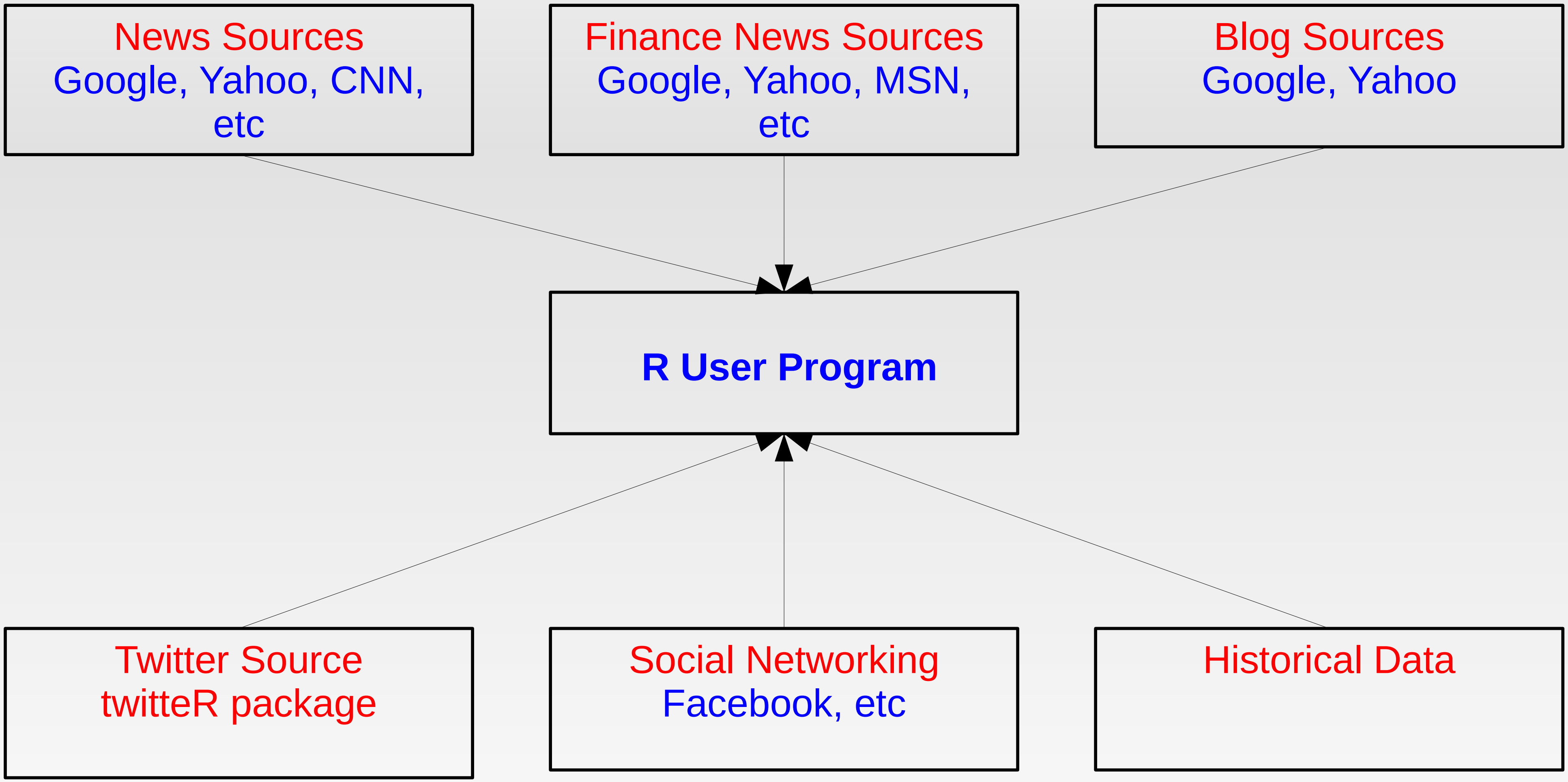
```
> unlist(strsplit(Dostoevsky[[1]][(start+6):(start+8)], split="[[:space:]]", perl=T))
 [1] "On"          "an"            "exceptionally" "hot"
 [5] "evening"     "early"         "in"            "July"
 [9] "a"           "young"         "man"           "came"
[13] "out"         "of"            "the"           "garret"
[17] "in"          "which"         "he"            "lodged"
[21] "in"          "S."            "Place"         "and"
[25] "walked"      "slowly,"       "as"            "though"
[29] "in"          "hesitation,"   "towards"       "K."
```

Using tm Plugins

- The tm package has become very popular in a number of industries – academics, bioinformatics, linguistics, finance, actuary, news analysis, etc.
- Number of people have developed plugin for tm package.
  eg: tm.plugin.webmining, tm.plugin.sentiment, tm.plugin.dc (for distributed computing)

News Aggregation Using R Packages

How to gather news using R



Read News using R

- Let's find out what's going on at SMU.

```
> corpus<-WebCorpus(GoogleNewsSource("Southern Methodist University"))
> corpus[[1]]
SMU's Engaged Learning Day inspires student projects
Email: brekow@smu.edu
Published: Monday, February 13, 2012
```

Headlines:

an@vaio: ~/courses/AdvDM/TextMining

```
> sapply(corpus, function(x) {attr(x, "Heading")})
  [1] "SMU's Engaged Learning Day inspires student projects - The Daily Campus"

  [2] "Journalism graduate nominated for four Emmy awards - The Daily Campus"

  [3] "Senate candidate can't escape sports scandals - The Associated Press"

  [4] "SMU sends 18 to Midwest LGBT conference - Dallas Voice"

  [5] "Baylor School of Music Lyceum Series Welcomes Art Historian for Lecture - Baylor University"
  [6] "Brooke Reyes receives bachelor's degree from Southern Methodist University - Your Houston News"
  [7] "Review: New York Baroque Dance Company | Dallas Bach Society - TheaterJones Performing Arts News in North Texas"
  [8] "College notes: 02.13.12 - Corpus Christi Caller Times"

  [9] "In Albania, Can a US Diploma Deliver? - New York Times"

 [10] "Closure Takes Top Spot at 2012 Indie Game Challenge - MarketWatch (press release)"
 [11] "Senate candidate can't escape sports scandals - Houston Chronicle"
```

Financial News

- GoogleFinanceSource can be used to get latest news about any listed company

  Example:

  corpus <- WebCorpus(GoogleFinanceSource("NASDAQ:MSFT"))

  Retrives news stories from Google about Microsoft Corporation and creates a corpus.

  corpus <- WebCorpus(YahooFinanceSource("MSFT"))

  Retrives news stories from Yahoo about Microsoft Corporation and creates a corpus.

corpus <- WebCorpus(TwitterSource("Microsoft"))

Retrives news stories from Twitter about Microsoft Corporation and creates a corpus.

Financial Data

- R package quantmod can be used to obtain latest stock market data.

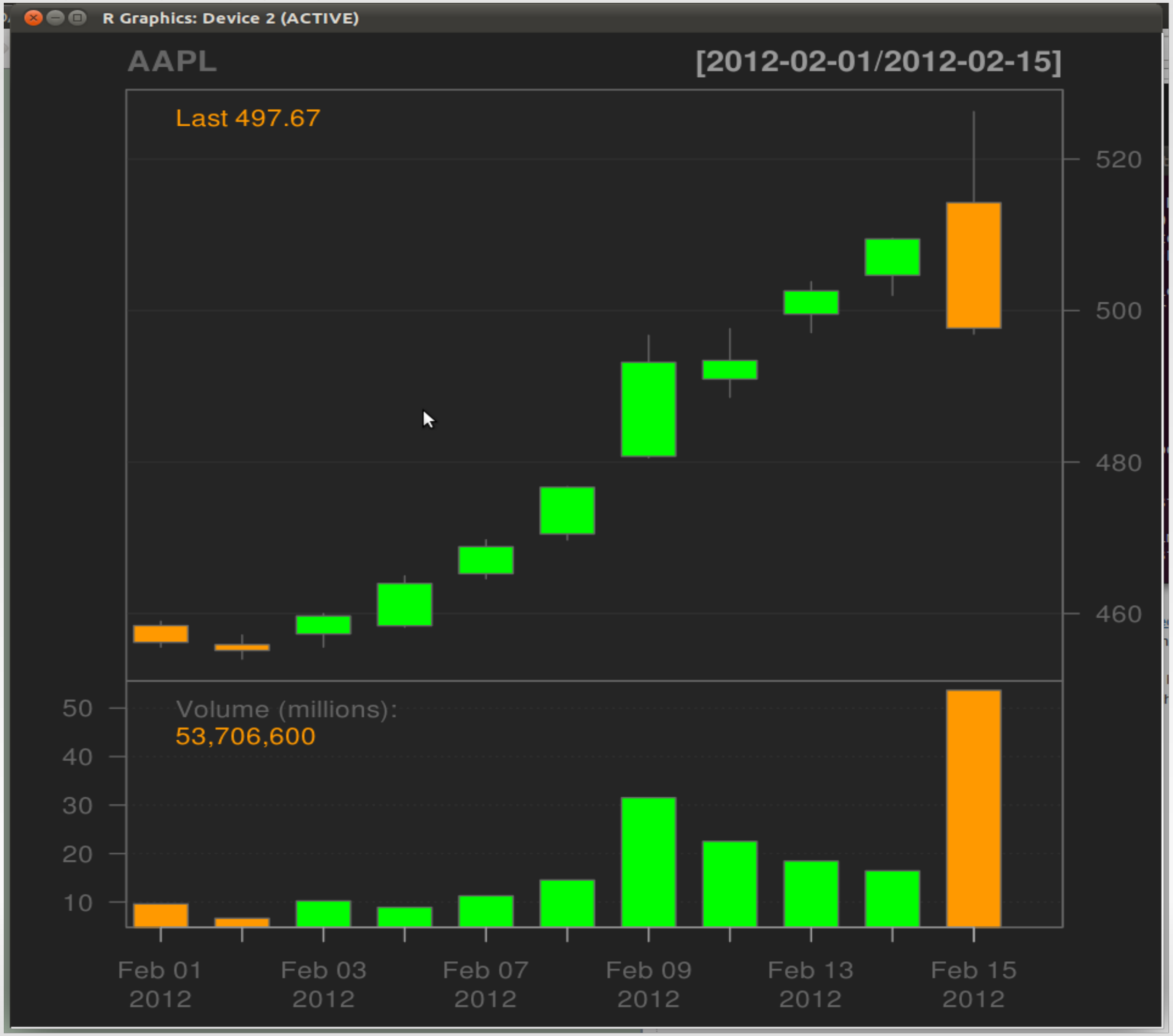


Chart of Apple (NASDAQ:AAPL) for month of February

Financial Data

- Can also download data as a matrix

an@vaio: ~/development/mmsa/pkg

```
> last(AAPL,n=10)
           AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume AAPL.Adjusted
2012-02-02    455.90    457.17   453.98     455.12     6661100        455.12
2012-02-03    457.30    460.00   455.56     459.68    10235700        459.68
2012-02-06    458.38    464.98   458.20     463.97     8907600        463.97
2012-02-07    465.25    469.75   464.58     468.83    11280600        468.83
2012-02-08    470.50    476.79   469.70     476.68    14544700        476.68
2012-02-09    480.76    496.75   480.56     493.17    31527700        493.17
2012-02-10    490.96    497.62   488.55     493.42    22523900        493.42
2012-02-13    499.53    503.83   497.09     502.60    18454300        502.60
2012-02-14    504.66    509.56   502.00     509.46    16442800        509.46
2012-02-15    514.26    526.29   496.89     497.67    53706600        497.67
>
```

Financial News

- Let's get news about Apple Corp.

```
> corpusAAPL <- WebCorpus(GoogleFinanceSource("NASDAQ:AAPL"))
> sapply(corpusAAPL, function(x) {attr(x, "Heading")})
 [1] "Apple&#39;s latest PC OS gets even more iOS-like"

 [2] "OS X 10.8 Mountain Lion Growls at the Masses"

 [3] "Apple Reverses, Stocks Top Out"

 [4] "Dominant Apple Looks To Cripple Rival Samsung"

 [5] "Amazon Declines on Morgan Stanley Downgrade"

 [6] "Apple&#39;s 4Q Global Tablet Market Share Falls To 57% From 64% In 3Q"

 [7] "Apple responds to furor over info-stealing apps"

 [8] "Apple Share Run Paused"
```

How to get News Sentiment

- How can we **automatically** analyze news and get a feel whether it conveys positive or negative sentiments.

News Sentiment

- R package tm.plugin.tags can help us.
- Contains large listing of positive and negative terms that can be used to tag news items.

```
> require("tm.plugin.tags")
> control <- list(stemming = TRUE)
> sample(tm_get_tags("Negativ", control = control), 10)
 [1] "gloomi"       "muddi"        "betray"       "disprov"      "substitut"
 [6] "unnecessari" "cannib"       "hazi"         "undon"        "burn"
> sample(tm_get_tags("Positiv", control = control), 10)
 [1] "humbl"        "fortun"       "spectacular" "respons"      "promin"
 [6] "hilari"       "glad"         "versatil"     "pleasant"     "golden"
```

News Tagging

Let's see which terms are tagged as positive in news for Apple Corp.

```
> colnames(AAPL_dtm_reduced)[which(which_pos==TRUE)]
  [1] "accept"      "accord"      "adjust"      "admit"       "agreement"
  [6] "aid"         "allow"       "appeal"      "approach"    "asset"
 [11] "attract"     "basic"       "benefit"     "board"       "bolster"
 [16] "boost"       "call"        "common"      "confer"      "consent"
 [21] "contact"     "content"     "correct"     "credit"      "deal"
 [26] "discuss"     "entertain"   "enthusiasm"  "establish"   "excel"
 [31] "fair"        "familiar"    "favor"       "forward"     "free"
 [36] "fresh"       "friend"      "gain"        "game"        "glow"
 [41] "gold"        "grace"       "hand"        "haven"       "health"
 [46] "help"        "hit"         "home"        "hope"        "hug"
 [51] "impress"     "inform"      "joke"        "keen"        "kid"
 [56] "law"         "lead"        "legal"       "live"        "loyal"
 [61] "main"        "major"       "matter"      "meet"        "offer"
 [66] "offset"      "partner"     "pass"        "patient"     "permit"
 [71] "plain"       "popular"     "premium"     "prime"       "pro"
 [76] "profit"      "progress"    "protect"     "real"        "regard"
 [81] "respect"     "return"      "rich"        "robust"      "round"
 [86] "safe"        "serious"     "share"       "smile"       "smitten"
 [91] "sought"      "special"     "stand"       "straight"    "success"
 [96] "suit"        "support"     "talent"      "thank"       "tradition"
[101] "travel"      "true"        "truth"       "understand"  "uphold"
```

News Tagging

Let's see which terms are tagged as negative in news for Apple Corp.

```
> colnames(AAPL_dtm_reduced)[which(which_neg==TRUE)]
  [1] "argument"   "attack"     "avoid"      "bankrupt"   "bar"
  [6] "barrier"    "beat"       "bit"        "blame"      "block"
 [11] "board"      "boot"       "box"        "break"      "broke"
 [16] "cancel"     "cancer"     "chaotic"    "close"      "combat"
 [21] "commit"     "compel"     "competitor" "complaint"  "concern"
 [26] "contend"    "cost"       "covert"     "critic"     "cross"
 [31] "crude"      "cut"        "danger"     "deal"       "death"
 [36] "default"    "deficit"    "difficult"  "disrupt"    "doubt"
 [41] "drive"      "drop"       "engulf"     "evil"       "fail"
 [46] "fall"       "fear"       "fed"        "fight"      "fire"
 [51] "flaw"       "hand"       "hard"       "help"       "hit"
 [56] "hoard"      "hole"       "hot"        "hurt"       "kill"
 [61] "lack"       "limit"      "lone"       "lose"       "loser"
 [66] "loss"       "lost"       "low"        "lower"      "matter"
 [71] "mean"       "mischief"   "mix"        "object"     "odd"
 [76] "pass"       "poor"       "pretend"    "push"       "quit"
 [81] "ration"     "reject"     "retreat"    "rival"      "run"
 [86] "scare"      "sever"      "short"      "sick"       "spite"
 [91] "spot"       "stick"      "stress"     "struck"     "succumb"
 [96] "suffer"     "threaten"   "try"        "undo"       "upset"
[101] "wait"       "war"        "withheld"   "woe"
```

Problems with first approach

1. Large number of terms – some irrelevant to stock price prediction.
2. The entire news story is not relevant for us. We need short meaningful ideas.
3. Need to explore some new way to get relevant information.
4. Much of the news story consists of analysis, comparison, past performance reviews => which is not really useful for present analysis.

Using just the Headlines

- Headlines are more meaningful and relevant.
- Terms can be better mined.
- Can give weights to terms based on the fluctuations they cause. For example, if there is a rise of 5% in stock price, I will give a weight of 5% to each of the positive terms.
- The weights will average out over time and for various companies

Headlines Analysis

Used 100 news articles to extract headlines for AAPL

an@vaio: ~/development/mmsa/pkg

```
> colnames(dtm_reduced)[which(which_pos==TRUE)]
 [1] "agreement" "appeal"    "art"       "award"     "basic"     "board"
 [7] "buy"       "correct"   "deal"      "discuss"   "entertain" "eye"
[13] "favor"     "hand"      "home"      "join"      "lead"      "legal"
[19] "protect"   "round"     "share"     "suit"      "uphold"
> colnames(dtm_reduced)[which(which_neg==TRUE)]
 [1] "antitrust" "attack"    "bankrupt"  "block"     "blow"      "board"
 [7] "cheap"     "complaint" "deal"      "delay"     "drop"      "fall"
[13] "fix"       "hand"      "hoard"     "limit"     "lone"      "lose"
[19] "lower"     "pound"     "reject"    "rival"     "skirmish"  "vie"
[25] "war"       "wrong"
>
```

More negative words, which are strong. Stock is likely to move downward.

Headlines Analysis

Possible sentiment metrics:

- Ratio of Number of Positive to Negative Terms:

  Sentiment = Number of Positive / Number of Negative terms

- Weighting Scheme:

  Find weights for positive and negative terms using long term average.
  For example, w(bankrupt) = -0.10, w(profit)=0.03, etc
  Using this, compute average weight of all terms and predict stock price movement.

Headlines Analysis

Project Plan:

- Gather News Terms for day x and stock prices for day x+1 for a chosen few companies for various sectors – tech, financial, energy, banks, etc.
- Find correlation between ratio of positive to negative terms to stock price change.
- Assign weights to terms based on direction and amount of change i.e. if price goes up, assign weight to positive terms and vice versa.
- Use the weights and correlation to predict stock prices for test dataset.

Conclusion

- Text Mining is perhaps the most important and ubiquitous area of data mining.
- Ever growing mass of data needs to be processed efficiently and analyzed.
- Text data presents unique challenges.
- Need for high performance tools and faster algorithms.
- Need for automated information extraction from vast text resources.
- I presented a case study of automatically extracting sentiment from news and using it for stock market prediction.

References

1. Ingo Feinerer (2012). **tm: Text Mining Package**. *R package version 0.5-7.1*
2. Ingo Feinerer, Kurt Hornik, and David Meyer (2008). **Text Mining Infrastructure in R.** *Journal of Statistical Software 25/5.*
3. Theussl, S. et al (2010). **Distributed Text Mining with tm.** *R Finance Conference.*
4. Theussl, S. et al (2010). **Do Media Sentiments Reflect Economic Indices.** *Statistical Learning Notes, Vienna University of Economics and Business.*
5. Ingo Feinerer. **An introduction to text mining in R**. *R News*, 8(2):19-22, October 2008.

References

6. Yu, W-B. et al (2007). **A theoretical framework integrating Text Mining and Energy Demand Forecasting**. *Intl Journal of Electronic Business Management* 5(3) 211-224
7. W. Zhang and S. Skiena. **Trading strategies to exploit blog and news sentiment**. *In The 4th International AAAI Conference on Weblogs and Social Media*, 2010.
8. Breen, J.(2011). **R by example: mining Twitter for consumer attitudes towards airlines.** *Boston Predictive Analytics MeetUp Presentation.*
9. Kowalczyk, W.(2006). **Introduction to Text Mining**. *Machine Learning Day www.cs.vu.nl*
10. Sanderson, R.(2008). **COMP527 Data Mining.** *Department of Computer Science, University of Liverpool.*

References

11. Grobelnik, M. **Text Mining Tutorial.** ***http://ailab.ijs.si/marko_grobelnik/***

12. Ananiadou, S. **What is Text Mining**. *National Center for Text Mining www.nactem.ac.uk*

13. Wikipedia, **Text Analytics** http://en.wikipedia.org/wiki/Text_analytics