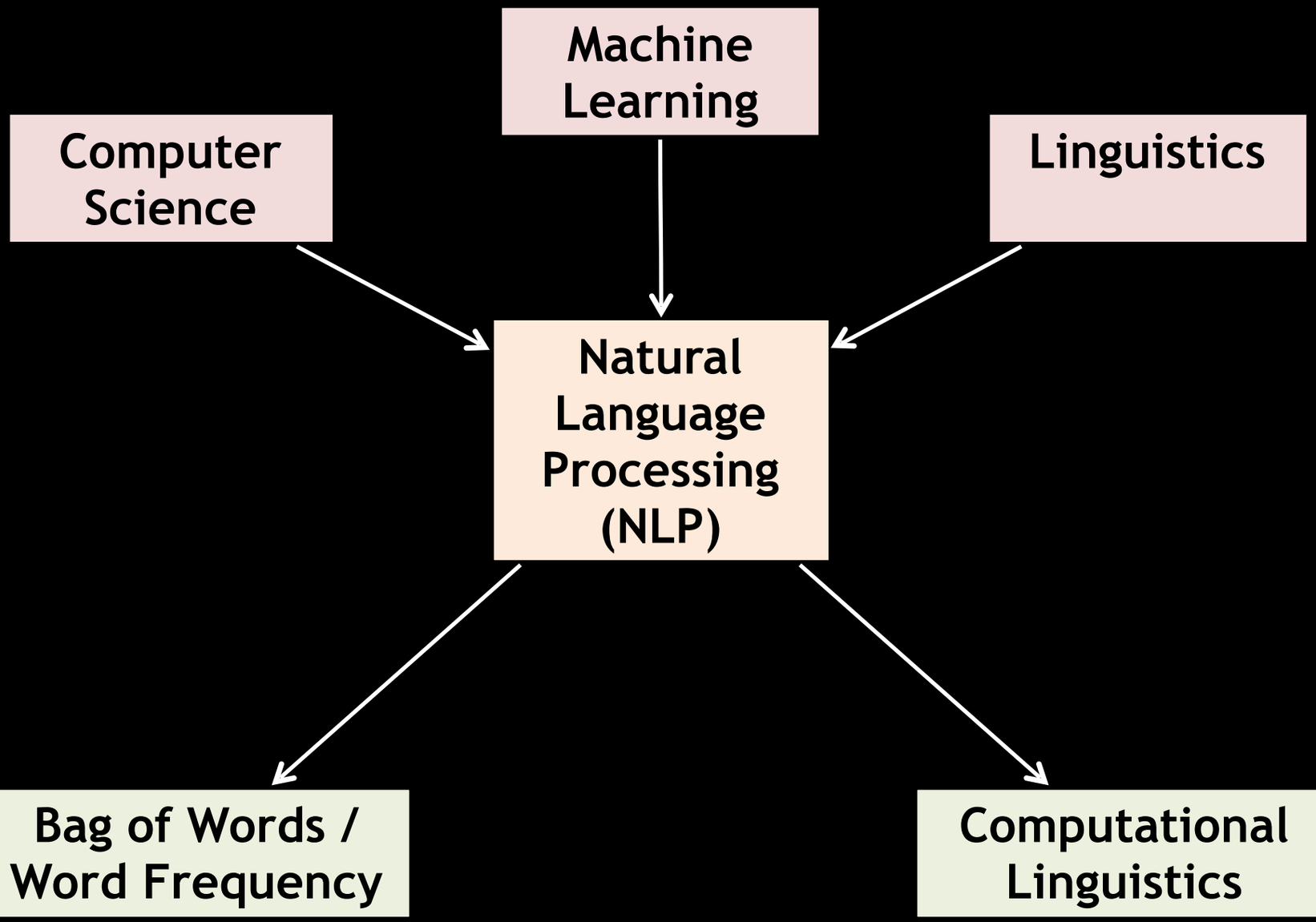
# Natural Language Processing in R

Thomas W. Jones

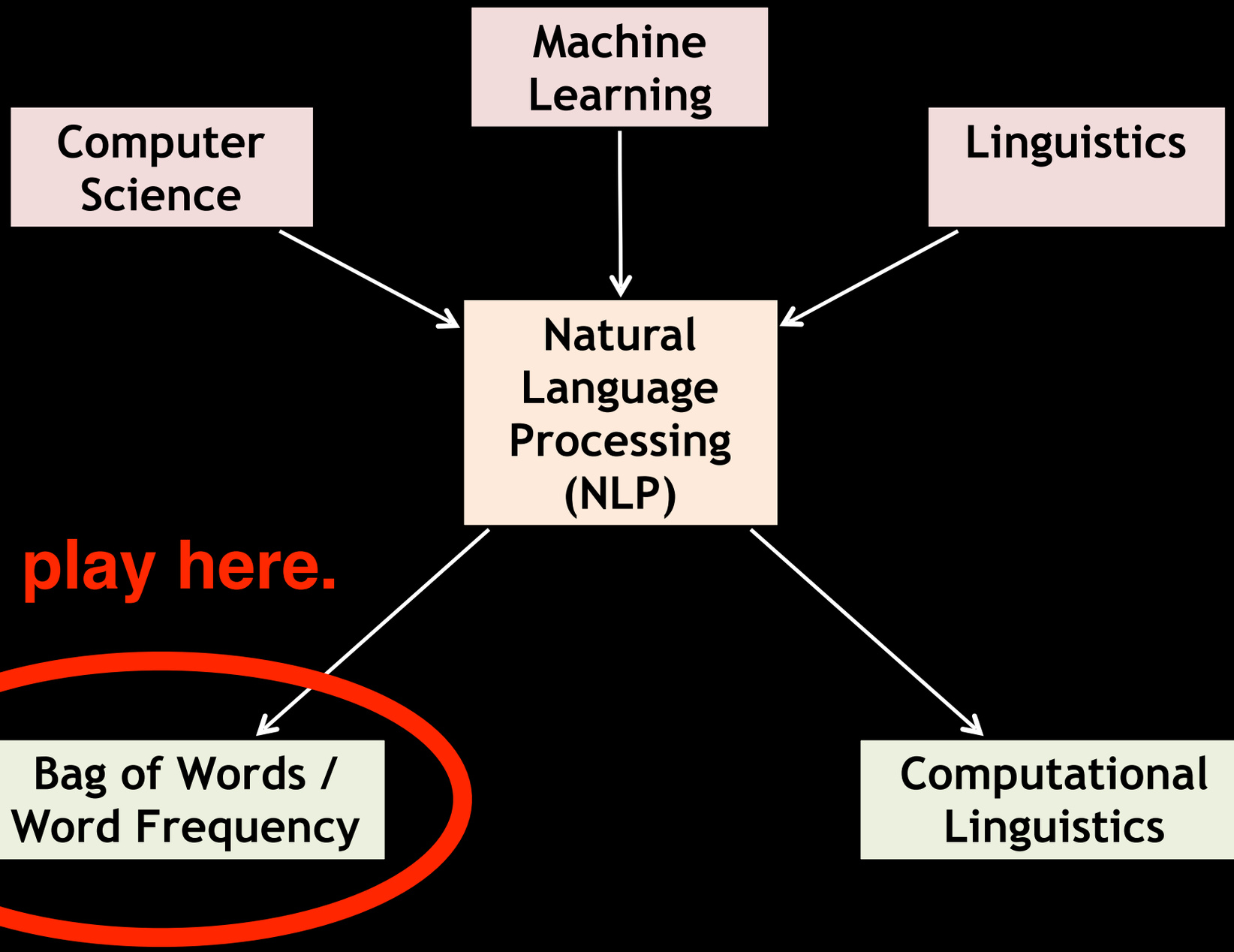
Delivered at Statistical Programming DC July 9, 2014

TOMMY DO MATH!

# What is NLP?


Machine Learning
Computer Science
Linguistics
Natural Language Processing (NLP)
Bag of Words / Word Frequency
Computational Linguistics

# What is NLP?

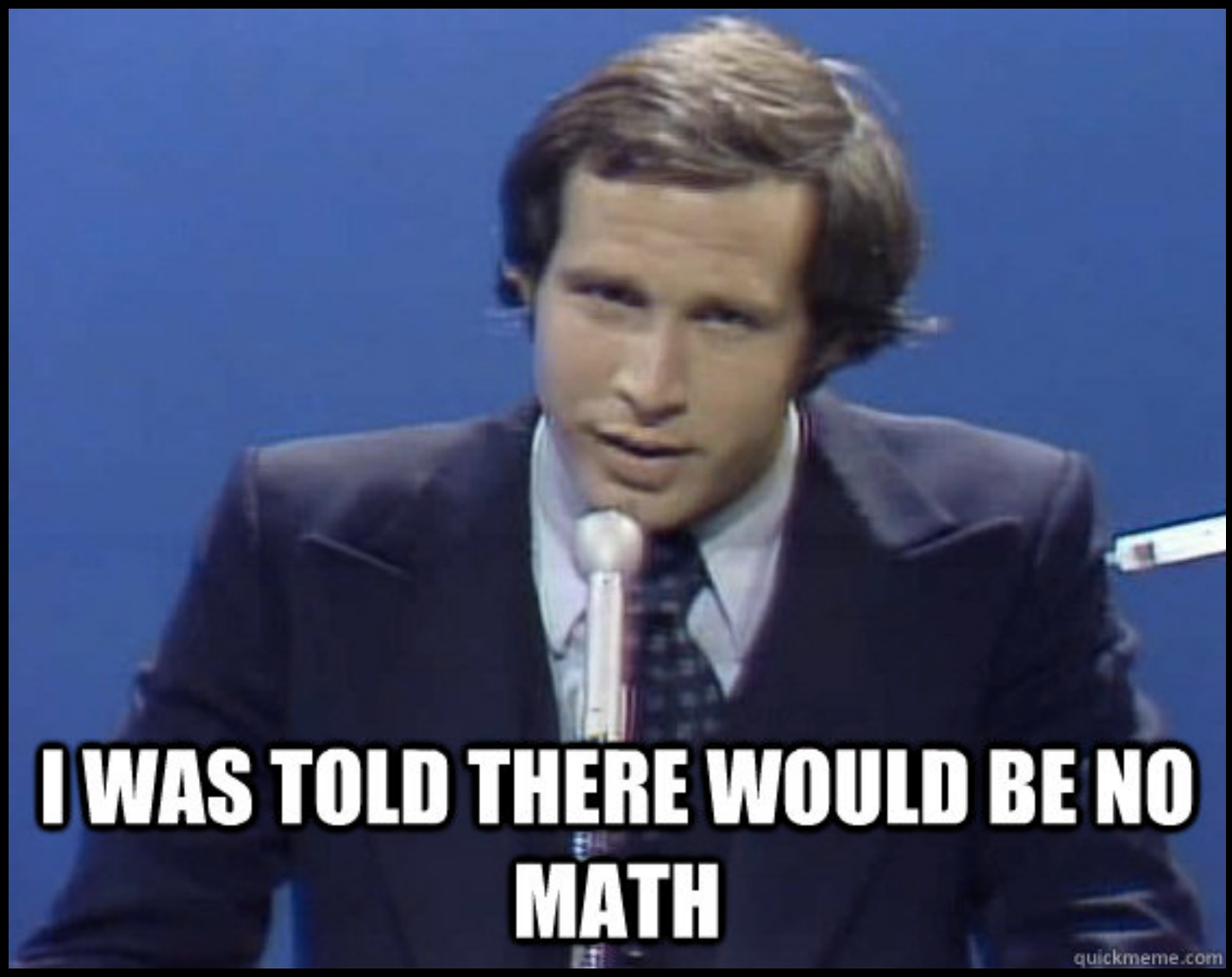

Machine Learning
Computer Science
Linguistics
Natural Language Processing (NLP)
I play here.
Bag of Words / Word Frequency
Computational Linguistics

# Why R?

“R is a language that was designed specifically for statistical computing and graphics. As a result, it has a wide following in the statistics community, which has in turn created a very extensive set of packages for ML & statistics.”

- Christopher Hefele
- (https://www.kaggle.com/forums/t/5243/pros-and-cons-of-r-vs-python-sci-kit-learn/39219)

# Who is This Talk For?




http://www.madmann.com/content/wp-content/uploads/2012/10/nomath.jpg

# Agenda

- Getting Started
- The Document Term Matrix
- Task 1: Document Clustering
- Task 2: A Basic Document Summarizer
- Task 3: A Basic Keyword Extractor
- CRAN Task View of NLP

# Agenda

- **Getting Started**
- The Document Term Matrix
- Task 1: Document Clustering
- Task 2: A Basic Document Summarizer
- Task 3: A Basic Keyword Extractor
- CRAN Task View of NLP

# With R You Have…

- **NLP Frameworks**
  - OpenNLP, tm, Weka, more…
- **Lexical dictionaries & Algorithms**
  - OpenNLP, WordNet, Porter's word stemmer, more…
- **Modeling algorithms**
  - Topic models, LSA, SVMs, clustering, classification, more…

# R has some sharp edges.

- Entire workspace held in RAM
- Parallelization is not automatic
- R is optimized for vectorized operations (matrix algebra)

# R has some nice features.

- Memory: bigmemory, bigalgebra, etc.
- Parallel processing: snow, snowfall, parallel, etc…
- Access other languages: rJava, rPython, rCpp, etc…

# Getting Started: base R functions for strings

- grep(), grepl(), gsub(), %in%, substr(), strsplit(), paste(), tolower(), toupper(),…..

- Use (learn) regular expressions!
  - http://www.rexegg.com/

# Our Example Corpus

- 68 Wikipedia articles from 3 Wikipedia categories:
  - Natural Language Processing
    - http://en.wikipedia.org/wiki/Category:Natural_language_processing
  - Statistical NLP
    - http://en.wikipedia.org/wiki/Category:Statistical_natural_language_processing
  - Computational Linguistics
    - http://en.wikipedia.org/wiki/Category:Computational_linguistics

# Agenda

- Getting Started
- **The Document Term Matrix**
- Task 1: Document Clustering
- Task 2: A Basic Document Summarizer
- Task 3: A Basic Keyword Extractor
- CRAN Task View of NLP

# The Document Term Matrix

http://web.presby.edu/~wasmith/newcourses/458busintel/notes/pix/fig05_06.jpg

| Documents \ Terms | Investment Risk | Project Management | Software Engineering | Development | SAP | . . . |
|---|---|---|---|---|---|---|
| Document 1 | 1 | | | 1 | | |
| Document 2 | | 1 | | | | |
| Document 3 | | | 3 | | 1 | |
| Document 4 | | 1 | | | | |
| Document 5 | | | 2 | 1 | | |
| Document 6 | 1 | | | 1 | | |
| . . . | | | | | | |

Copyright © 2011 Pearson Education, Inc. publishing as Prentice Hall

# Decisions Decisions…

- What will my columns be? (terms)
- What will my rows be? (chunks of text)
- What will the entries of my matrix be? (frequency measure)

# What Will my Columns Be?

- Do I remove stopwords? Which ones?
- What do I do with punctuation? Numbers?
- Where does each word start and end? (*tokenization*)
- Do I lemmatize? Do I stem?
- Do I want single words or n-grams?

# What will my rows be?

- Whole documents? (Articles, pages, books, etc.)
- Parse documents by chapter? Pages? Sentences?
- Aggregate small documents?

# What Will my Matrix Entries Be?

- Integer or “raw” counts of words?
- Normalized by document length?
- Frequency-weighted? (e.g. TF-IDF)

# Keeping it Simple Today

- **Columns:**
  - Single words with stop words and low-frequency words removed (no numbers/punctuation either)
- **Rows:**
  - Documents for clustering, sentences for summarization and keyword extraction
- **Frequency:**
  - TF-IDF for clustering, “raw” counts for summarization and keyword extraction

# In R: Read in Docs

```
documents <- dir("data/WikiSubset/") # get document names

getDocs <- function(DOC){
    result <- scan(paste("data/WikiSubset/", DOC, sep=""), what="character", sep="\n") # read file

        # do some cleanup
        result <- gsub("<[^<>]*>|\\{[^\\{\\}]*\\}|\"", "", result) # remove some html related things

        result <- result[ ! grepl("^$|^\\s+$", result) ] # keep only lines with text on them

        result <- gsub(" +|\t+", " ", result) # get rid of extra spaces and tabs

        result <- gsub("^ +| +$", "", result) # get rid of leading spaces and tabs

        result <- paste(result, collapse="\n") # collapse into a single-entry

        return(result)
}

library(snowfall)
sfInit(parallel=TRUE, cpus=4) # read in documents in parallel
sfExport("getDocs")
    documents <- sfSapply(documents, function(x) getDocs(DOC=x) )
sfStop()
```

# In R: Make DTM

```
library(tm)
############################
# Some Cleanup
############################
corp <- Corpus(VectorSource(documents)) # make a corpus object

corp <- tm_map(corp, tolower) # make everything lowercase

corp <- tm_map(corp, removeWords, c(stopwords("english"), stopwords("smart"))) # remove stopwords

corp <- tm_map(corp, removePunctuation) # remove punctuation

corp <- tm_map(corp, removeNumbers) # remove numbers

corp <- tm_map(corp, stripWhitespace) # get rid of extra spaces

########################
# Make the DTM
########################
dtm.tm <- DocumentTermMatrix(x=corp, control=list(weighting=weightTf)) # dtm of raw word counts
```

# A Note on the “tm” Package

- Aims to be an “all-in-one” framework for text mining in R
- Most commonly-used framework for other NLP packages
- DocumentTermMatrix objects are “simple triplet” matrices from the “slam” package.
- Simple triplet matrices have different methods than normal matrices in R.
- Sparse matrices from the “Matrix” package have the same methods!

http://tm.r-forge.r-project.org/

# In R: Convert to DTM to a Sparse Matrix

```
library(slam)
library(Matrix)
##########################
# Conversion
##########################
vocab <- Terms(dtm.tm) # preserve our vocabulary
docnames <- Docs(dtm.tm) # preserve document names

dtm.Matrix <- sparseMatrix(i=dtm.tm$i, j=dtm.tm$j, x=dtm.tm$v,  # converts to a sparse dtm
                           dims=c(dtm.tm$nrow, dtm.tm$ncol))

colnames(dtm.Matrix) <- vocab
rownames(dtm.Matrix) <- docnames

dim(dtm.Matrix) # how big is this matrix?

##########################
# Drop Additional Terms
##########################

doc.freq <- colSums(dtm.Matrix > 0) # get document frequency

# drop terms in half or more of the documents or words that appear in 3 or fewer documents
dtm.Matrix <- dtm.Matrix[ , doc.freq < nrow(dtm.Matrix)/2 & doc.freq > 3 ]
```

# Memory Management and DTMs

```
> # preview
> dtm.Matrix[ 1:10, 1:10 ]
10 x 10 sparse Matrix of class "dgCMatrix"
   [[ suppressing 10 column names ‘ability’, ‘abstract’, ‘academic’ ... ]]

Abzooba.html                       . . . . . . . . . .
Acoustic_model.html                . . . . . . . . . .
Additive_smoothing.html            . . . . . . . . . .
BabelNet.html                      . . . . . . . . . .
Bag-of-words_model.html            . . . . . . 1 . . .
Cache_language_model.html          . . . . . . . . . .
Collostructional_analysis.html     . . . . . . . . . .
Computational_linguistics.html     6 1 1 . . . . . 1 2
DATR.html                          . . . . . . . . . .
DialogOS.html                      . . . . . . . . . .
```

```
>
> object.size(dtm.Matrix) # about 222 Kb
227808 bytes
>
> object.size( as.matrix(dtm.Matrix) ) # about 828 Kb, almost 4 times bigger
!
848016 bytes
>
```

# Do Not Do This!

"In the end MALLET spat out a 7GB file, a matrix just shy of 1 million rows by 1000 columns or so, which was much to large to read in to R. Or rather, I could read in the file, but not do much with it afterwards."

http://www.r-bloggers.com/scaling-up-text-processing-and-shutting-up-r-topic-modelling-and-mallet/

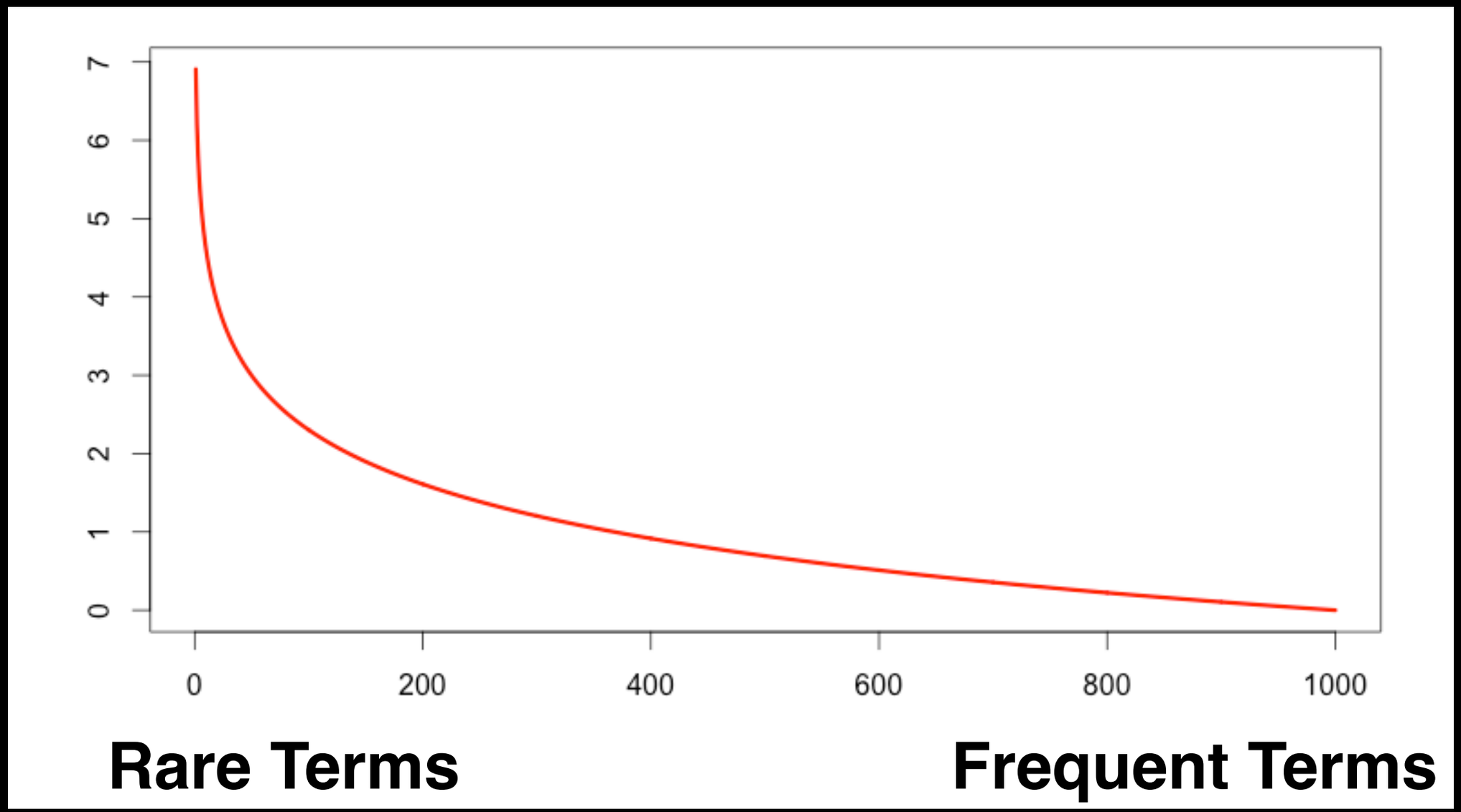
# Agenda

- Getting Started
- The Document Term Matrix
- **Task 1: Document Clustering**
- Task 2: A Basic Document Summarizer
- Task 3: A Basic Keyword Extractor
- CRAN Task View of NLP

# TF-IDF Frequency Conversion

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

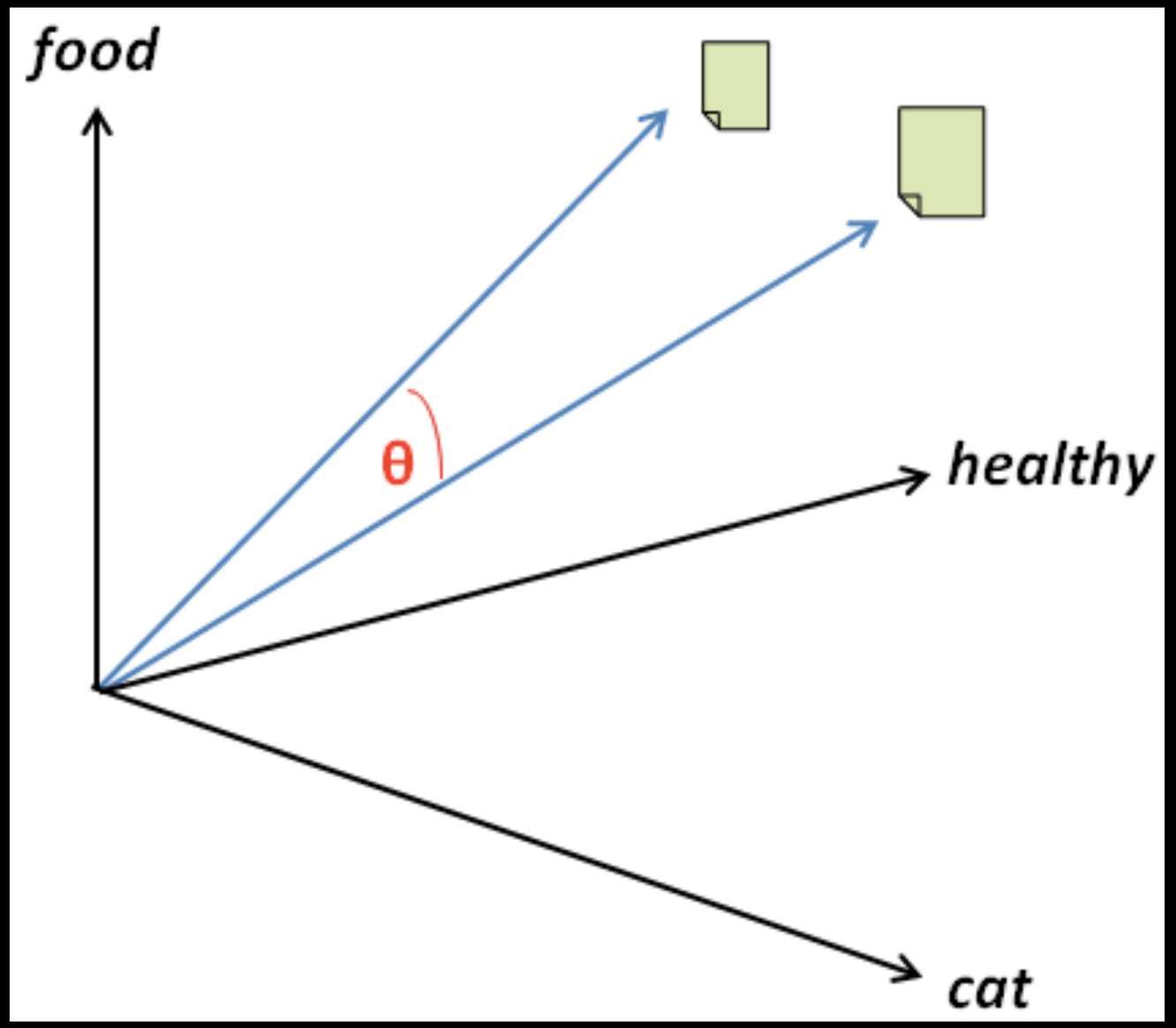
# IDF Weights



# In R: Make TF-IDF

```r
idf <- log(nrow(dtm.Matrix) / colSums(dtm.Matrix > 0))

dtm.Matrix <- dtm.Matrix / rowSums(dtm.Matrix) # normalize document length

dtm.Matrix <- t(dtm.Matrix) * idf # transpose for correct multiplication

dtm.Matrix <- t(dtm.Matrix) # get it back in the right format
```

# Distance $_{x,y}$ = 1 - Cosine $_{x,y}$




http://anythingbutrbitrary.blogspot.com/2013/03/build-search-engine-in-20-minutes-or.html

# In R: Get Cosine Distance

```r
# make every vector unit length
dtm.Matrix <- t(apply(dtm.Matrix, 1, function(x){
    x <- x / sqrt(sum( x * x ))
}))

# matrix multiplication is fast in R!
cosine.sim <- dtm.Matrix %*% t(dtm.Matrix)

# make me a dist() object
cos.dist <- as.dist( 1 - cosine.sim )
```

# In R: Hierarchical Clustering

```r
clustering <- hclust(cos.dist, method="ward.D")
clustering <- cutree(clustering, k=10) # 10 clusters is arbitrary
```

```
[1] "cluster  6"
[1] "DATR.html"                     "Eagles_Guidelines.html"          "Inside_Outside_Beginning.html"
[4] "JAPE_(linguistics).html"       "PATR-II.html"                    "Semantic_analysis_(computational).html"
[7] "Stochastic_grammar.html"
[1] "_____________________________________"
[1] "_____________________________________"
[1] "cluster  7"
[1] "DialogOS.html" "SABLE.html"
[1] "_____________________________________"
[1] "_____________________________________"
[1] "cluster  8"
[1] "Dissociated_press.html"                    "Gale%E2%80%93Church_alignment_algorithm.html" "Markov_information_source.html"

[4] "Noisy_channel_model.html"                  "T9_(predictive_text).html"                    "Tatoeba.html"

[1] "_____________________________________"
[1] "_____________________________________"
[1] "cluster  9"
[1] "History_of_machine_translation.html"       "Interactive_machine_translation.html"       "Machine_translation_software_usability.h
tml"
[4] "Machine_translation.html"
[1] "_____________________________________"
[1] "_____________________________________"
[1] "cluster  10"
[1] "Kaldi_(software).html"       "Natural_Language_Toolkit.html" "OpenNLP.html"
[1] "_____________________________________"
```

# Agenda

- Getting Started
- The Document Term Matrix
- Task 1: Document Clustering
- **Task 2: A Basic Document Summarizer**
- Task 3: A Basic Keyword Extractor
- CRAN Task View of NLP

# A Different Document Term Matrix

http://web.presby.edu/~wasmith/newcourses/458busintel/notes/pix/fig05_06.jpg

| Terms / Documents | Investment Risk | Project Management | Software Engineering | Development | SAP | . . . |
|---|---|---|---|---|---|---|
| Document 1 | 1 | | | 1 | | |
| Document 2 | | 1 | | | | |
| Document 3 | | | 3 | | 1 | |
| Document 4 | | 1 | | | | |
| Document 5 | | | 2 | 1 | | |
| Document 6 | 1 | | | 1 | | |
| . . . | | | | | | |

Copyright © 2011 Pearson Education, Inc. publishing as Prentice Hall

# A Different Document Term Matrix

http://web.presby.edu/~wasmith/newcourses/458busintel/notes/pix/fig05_06.jpg

| Documents \ Terms | Investment Risk | Project Management | Software Engineering | Development | SAP | . . . |
|---|---|---|---|---|---|---|
| Sentence 1 | 1 | | | 1 | | |
| Sentence 2 | | 1 | | | | |
| Sentence 3 | | | 3 | | 1 | |
| Sentence 4 | | 1 | | | | |
| Sentence 5 | | | 2 | 1 | | |
| Sentence 6 | 1 | | | 1 | | |
| . . . | | | | | | |

Copyright © 2011 Pearson Education, Inc. publishing as Prentice Hall

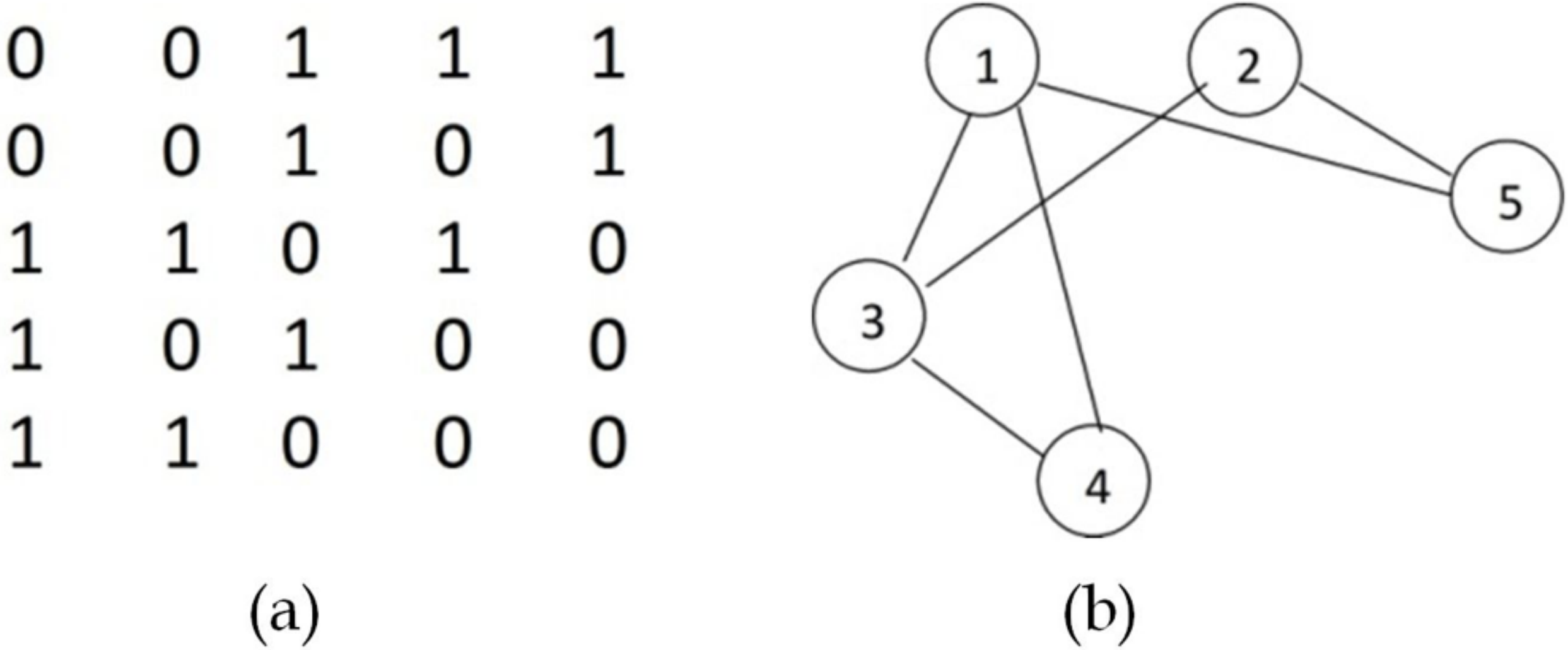
# In R: A Sentence Parser

```r
library(openNLP)
library(NLP)

ParseSentences <- function(doc){
    ##########################################################
    # Takes a single document, stored as an entry of a
    # character vector, and parses out the sentances.
    ##########################################################

    annotator <- Maxent_Sent_Token_Annotator(language = "en") # uses the Apache OpenNLP
                                                              # Maxent sentence detector

    doc <- as.String(doc) # convert format for compatibility with OpenNLP

    sentence.boundaries <- annotate(doc, annotator)

    result <- doc[ sentence.boundaries ] # extract those sentences!

    # if there aren't already names
    # name each sentence
    if( is.null(names(result)) ) names(result) <- paste("sen", 1:length(result), sep=".")

    return(result)
}
```

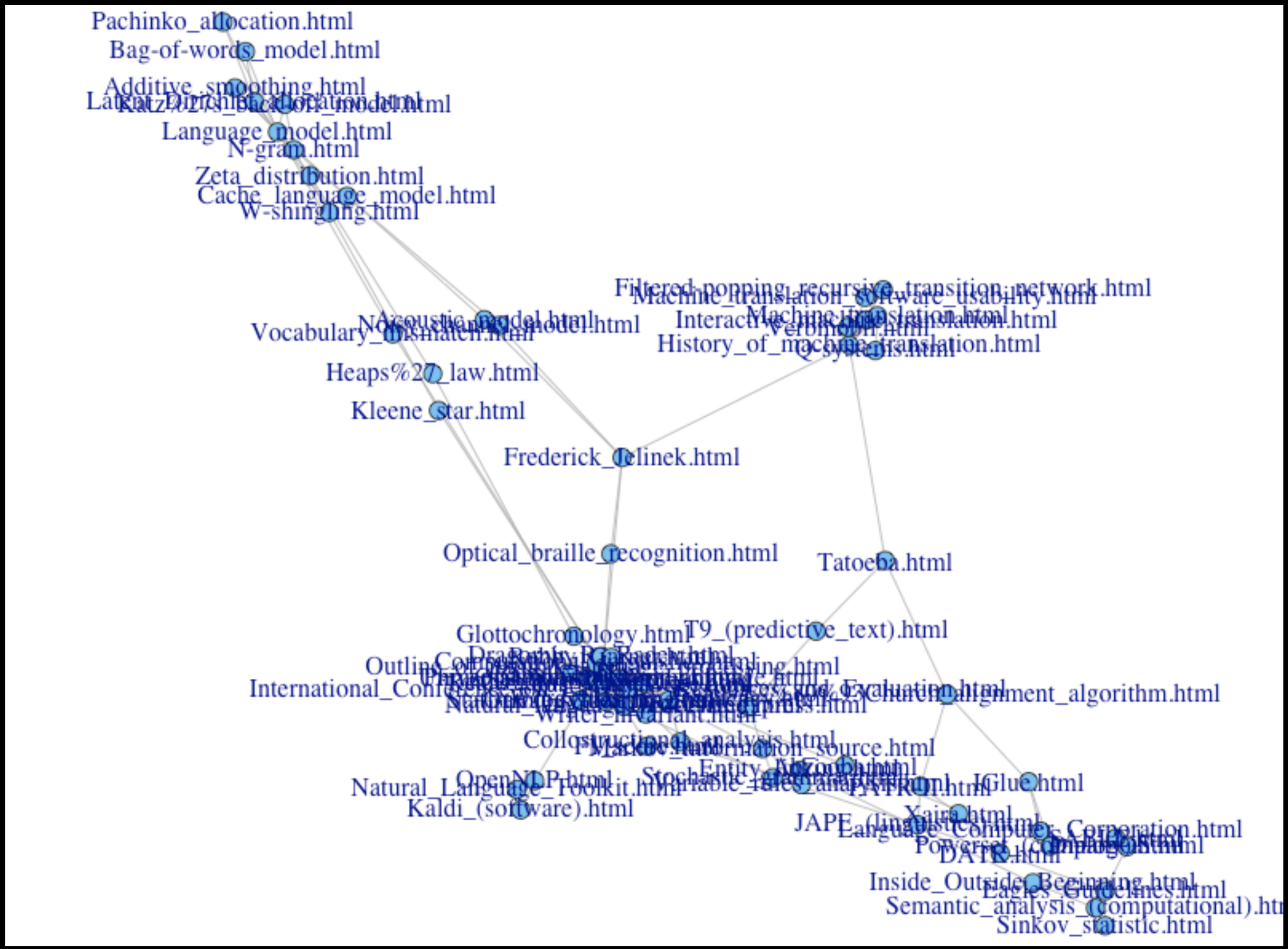
# In R: DTM-Making Function

```r
MakeSentenceDtm <- function(doc){
    ##########################################################
    #   doc = character vector whose entries
    #         correspond to sentences in a document
    ##########################################################

    ### pre-DTM cleanup
    doc <- gsub("[^a-zA-Z]", " ", doc) # removes all non-alphabetic characters
    doc <- tolower(doc) #lowercase
    doc <- gsub(" +", " ", doc) # removes extra spaces
    corp <- Corpus(VectorSource(doc))
    corp <- tm_map(corp, removeWords, # remove stopwords
                   gsub("[^a-zA-Z]", " ", c(stopwords("english"), stopwords("SMART")) ) )
    corp <- tm_map(corp, stripWhitespace) # remove spaces again

    ### tm DTM
    dtm <- DocumentTermMatrix(corp)
    ### Matrix DTM
    dtm.sparse <- sparseMatrix(i=dtm$i, j=dtm$j, x=dtm$v,  # converts to a sparse dtm
                               dims=c(dtm$nrow, dtm$ncol))
    rownames(dtm.sparse) <- Docs(dtm)
    colnames(dtm.sparse) <- Terms(dtm)

    # keep only sentences that have at between 5 and 20 words
    # this is arbitrary and could be adjusted/improved
    dtm.sparse <- dtm.sparse[ rowSums(dtm.sparse) %in% 5:20, ]

    return(dtm.sparse)
}
```

# Similarity Matrix or Adjacency Matrix?




http://www.intechopen.com/source/html/38196/media/image1.jpeg


Pachinko_allocation.html
Bag-of-words_model.html
Additive_smoothing.html
Language_model.html
N-gram.html
Zeta_distribution.html
Cache_language_model.html
W-shingling.html
Vocabulary_mismatch.html
Heaps%27_law.html
Kleene_star.html
Frederick_Jelinek.html
Optical_braille_recognition.html
Tatoeba.html
T9_(predictive_text).html
Natural_Language_Toolkit.html
Kaldi_(software).html
IGlue.html
Xaira.html
DATR.html
Inside_Outside_Beginning.html
Sinkov_statistic.html

# In R: Cosine Similarity to igraph object

```r
library(igraph)

GetCsimGraph <- function(dtm){
    dtm <- t( apply(dtm, 1, function(x){
        x <- x / sqrt(sum(x*x))
        return(x)
    }))

    csim <- dtm %*% t(dtm)

    g <- graph.adjacency(adjmatrix=csim, mode="undirected", weighted=TRUE, diag=FALSE)

    return(g)
}
```

# In R: Top N Most-Central Sentences

```r
EvCentTopN <- function(g, N){
    # top N sentences (keywords) based on eigenvector centrality
    top.n <- evcent(graph=g)
    top.n <- names(top.n$vector)[ order(top.n$vector, decreasing=TRUE) ][ 1:N ]

    return(top.n)
}
```

# In R: Put it all Together

```
sfInit(parallel=TRUE, cpus=4)

sfLibrary(igraph)
sfLibrary(openNLP)
sfLibrary(NLP)
sfLibrary(tm)
sfLibrary(Matrix)

sfExport(list=c("ParseSentences", "MakeSentenceDtm", "GetCsimGraph", "EvCentTopN"))

summaries <- sfSapply(documents, function(DOC){
    DOC <- ParseSentences(doc=DOC) # parse sentences
    dtm <- MakeSentenceDtm(doc=DOC) # make a dtm

    # keep only sentences that have at between 5 and 20 words
    # this is arbitrary and could be adjusted/improved
    dtm <- dtm[ rowSums(dtm) %in% 5:20, ]


    csim <- GetCsimGraph(dtm=dtm) # get cosine similarity graph
    top4 <- EvCentTopN(g=csim, N=4) # get the top 4 sentences

    result <- paste(DOC[ top4 ], collapse="\n")

    return(result)
})

sfStop()
```

# How did we do?

```
> cat(summaries[ "Acoustic_model.html" ])
Speech recognition engines work best if the acoustic model they use was trained with sp
eech audio which was recorded at the same sampling rate/bits per sample as the speech b
eing recognized.
Thus for desktop speech recognition, the current standard is acoustic models trained wi
th speech audio data recorded at sampling rates of 16 kHz/16bits per sample.
As a general rule, a speech recognition engine works better with acoustic models traine
d with speech audio data recorded at higher sampling rates/bits per sample.
Therefore, for telephony based speech recognition, acoustic models should be trained wi
th 8 kHz/8-bit speech audio files.
>
```

```
> cat(summaries[ "Machine_translation.html" ])
First, machine translation is much faster than human translation.
Name Translation in Statistical Machine Translation Learning When to Transliterate.
Second, machine translation uses a much larger quantity of vocabulary than human transl
ation.
According to the nature of the intermediary representation, an approach is described as
 interlingual machine translation or transfer-based machine translation.
>
```

# Full Disclosure

```
> cat(summaries[ "Abzooba.html" ])
clicking \\Summarize\\, you agree to the terms of use for this wiki.\,\wgFlowTermsOfUse
CloseTopic\:\By
clicking \\Close topic\\, you agree to the terms of use for this wiki.\,\wgFlowTermsOfU
seReopenTopic\:\By
By using this site, you agree to the Terms of Use and Privacy Policy.
Text is available under the Creative Commons Attribution-ShareAlike License;
additional terms may apply.
>
```

```
> cat(summaries[ "Latent_Dirichlet_allocation.html" ])
\end
\ src=\//upload.wikimedia.org/math/1/e/9/1e90cc19d4044f0f21d2687bc424489c.png\
src=\//upload.wikimedia.org/math/1/8/c/18ca693eb10b219b5468b01de4f8d644.png\
\end
\ src=\//upload.wikimedia.org/math/1/7/5/17541563f2162eba129d911c3e0154fc.png\
\end
\ src=\//upload.wikimedia.org/math/b/6/5/b658fcc850a54839f3ed99b88135cb94.png\
>
```

# Agenda

- Getting Started
- The Document Term Matrix
- Task 1: Document Clustering
- Task 2: A Basic Document Summarizer
- **Task 3: A Basic Keyword Extractor**
- CRAN Task View of NLP

# Keyword Extraction is Like Summarization

http://web.presby.edu/~wasmith/newcourses/458busintel/notes/pix/fig05_06.jpg

| Terms / Documents | Investment Risk | Project Management | Software Engineering | Development | SAP | . . . |
|---|---|---|---|---|---|---|
| Sentence 1 | 1 | | | 1 | | |
| Sentence 2 | | 1 | | | | |
| Sentence 3 | | | 3 | | 1 | |
| Sentence 4 | | 1 | | | | |
| Sentence 5 | | | 2 | 1 | | |
| Sentence 6 | 1 | | | 1 | | |
| . . . | | | | | | |

Copyright © 2011 Pearson Education, Inc. publishing as Prentice Hall

# Keyword Extraction is Like Summarization

http://web.presby.edu/~wasmith/newcourses/458busintel/notes/pix/fig05_06.jpg

| Terms / Documents | Sentence 1 | Sentence 2 | Sentence 3 | Sentence 4 | Sentence 5 | Sentence 6 | . . . |
|---|---|---|---|---|---|---|---|
| Investment Risk | 1 | | | | | 1 | |
| Project Management | | 1 | | 1 | | | |
| Software Engineering | | | 3 | | 2 | | |
| Development | 1 | | | | 1 | 1 | |
| SAP | | | 1 | | | | |
| . . . | | | | | | | |

Copyright © 2011 Pearson Education, Inc. publishing as Prentice Hall

# Homework: Get Keywords

If you really get stuck:

https://github.com/TommyJones/DocSummarizer

# Agenda

- Getting Started
- The Document Term Matrix
- Task 1: Document Clustering
- Task 2: A Basic Document Summarizer
- Task 3: A Basic Keyword Extractor
- **CRAN Task View of NLP**

# Going further….

## CRAN Task View: Natural Language Processing

**Maintainer:** Fridolin Wild, Knowledge Media Institute (KMi), The Open University, UK
**Contact:** fridolin.wild at open.ac.uk
**Version:** 2014-06-08

Natural language processing has come a long way since its foundations were laid in the 1940s and 50s (for an introduction see, e.g., Jurafsky and Martin (2008): Speech and Language Processing, Pearson Prentice Hall). This CRAN task view collects relevant R packages that support computational linguists in conducting analysis of speech and language on a variety of levels - setting focus on words, syntax, semantics, and pragmatics.

In recent years, we have elaborated a framework to be used in packages dealing with the processing of written material: the package tm. Extension packages in this area are highly recommended to interface with tm's basic routines and useRs are cordially invited to join in the discussion on further developments of this framework package. To get into natural language processing, the cRunch service and tutorials may be helpful.

### Frameworks:

- tm provides a comprehensive text mining framework for R. The Journal of Statistical Software article Text Mining Infrastructure in R gives a detailed overview and presents techniques for count-based analysis methods, text clustering, text classification and string kernels.
- tm.plugin.dc allows for distributing corpora across storage devices (local files or Hadoop Distributed File System).
- tm.plugin.mail helps with importing mail messages from archive files such as used in Thunderbird (mbox, eml).
- tm.plugin.alceste allows importing text corpora written in a file in the Alceste format.

http://cran.r-project.org/web/views/NaturalLanguageProcessing.html

# Thank You

**Contact Info:**
twjones@ida.org (work)
jones.thos.w@gmail.com (personal)
@thos_jones (twitter)

**Read my blog!**
www.biasedestimates.com