# MET CS688 OL
# WEB ANALYTICS AND MINING

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

LIVE CLASSROOM 3 (LECTURE SESSION)

# Module 3 Objectives

- Demonstrate how to load text from a group of documents.
- Breakdown the steps required to pre-process text, by
  - Using the tm library in R to convert the documents into a corpus.
  - Grouping similar words together, and learning how to do this in documents that contain various punctuation, capitalization, and grammar schemes.
  - Creating an index of which documents include which words, or terms, appear in which documents.
- Discuss the basics of fuzzy-string matching.
- Implement the k Nearest Neighbors classifier in R. This supervised learning uses information from some existing documents (forming a training data set) to predict the topics of additional documents (forming a test data set).

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

# Module 3 - Text Mining

Content:

- Preprocessing and content extraction
- Searching and fuzzy string matching
- Clustering text
- Classification, categorization, and tagging

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- Consider an application that allows a user to enter query to search all the files on your computer that may contain the entered keyword.
- The first thing you would need to do is to transform all these files into a common format so that you only have to worry about one format internally.

- This process of standardizing the many different file types to a common text-based representation is called preprocessing. Preprocessing includes any steps that must be undertaken before inputting data into the library or application for its intended use, extracting content and metadata from common file formats such as MS Word and Adobe PDF.

# Text Mining (Text Analytics)

Most of today's data is unstructured (in a form of mixed media)

- text, images, audio data, etc.

The information in the data eventually is retrieved in a textual form – the way we communicate and express ourselves.

- The most optimal, most compact way of keeping the content of the information we use on a daily basis.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- The goal of text mining (text analytics) is to obtain (retrieve) the essential information contained in the text data by using statistical pattern learning.
  - Goal - content management and information organization
  - Tools - applying the knowledge discovery to tasks such as
    - topic detection
    - phrase extraction
    - document grouping, etc.

# Text Mining (Text Analytics)

- Typical steps involved in information retrieval
  1. Structuring the input text
     - Preprocessing, parsing, searching for some linguistic features and the removal of others.
  2. Obtaining and matching patterns within the structured data (text).
     - Linear algebra, Machine Learning (regression, classification, clustering).
  3. Evaluation and interpretation of the results.
     - Statistics (Precision, Recall, F score etc.).

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

# Introduction

Example:

- We have a large set of digital text documents (books, newspapers, emails, etc.).
- We want to extract useful information from this massive amounts of text quickly.
- To summarize the main themes and identify those of most interest to us or to particular people (clients).
- Using automated algorithms, we can achieve this much quicker than a human could.
- To this we refer to as knowledge distillation:
  - Text classification/categorization
  - Document clustering—finding groups of similar documents
  - Information extraction
  - Summarization

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

# Text processing

- Why is Processing Text important?
  - All of our digital communication, language, documents, storage of knowledge etc. is in a textual form.
  - Understanding the information content of this data is part of our daily job.
  - Increase productivity, find relevant information quicker etc.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- Levels of text processing, by increasing complexity:
  - Characters
  - Words
  - Multiword text and sentences
  - Multi sentence text and paragraphs
  - Documents
  - Multi document text and corpora

# What Makes Text Processing Difficult?

- Challenges at many levels, from handling character encodings to inferring meaning.
- On a small scale, much easier to deal with:
  - Search the text data based on user input and return the relevant passage (a paragraph, for example).
  - Split the passage into sentences.
  - Extract “interesting” things from the text, for example the names of people.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- On a large scale, much harder to deal with:
  - Linguistics as syntax (grammar), understanding the rules about categories of words and word groupings.
  - Language translation. Have you tried Google translate?
  - “Understanding” the text content like people do.

- We are far from the famous Turing Test—a test to determine whether a machine's intelligent behavior is distinguishable from that of a human.

# Commonly Used Terms in Text Mining

- **Index** – This refers to cataloging (indexing) the documents in advance, to avoid linearly scanning the texts for each query.
- **Terms** – Terms are the indexed units (words) that we are searching for.
- **Term-Document Incidence Matrix** – This is a (binary) table that relates the terms and the documents. The transposed version is called **Document-Term Incidence Matrix.**

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

| Documents / Terms | Antony and Cleopatra | Julius Caesar | The Tempest | Hamlet | Othello | Macbeth | … |
|---|---|---|---|---|---|---|---|
| Antony | 1 | 1 | 0 | 0 | 0 | 1 | |
| Brutus | 1 | 1 | 0 | 1 | 0 | 0 | |
| Caesar | 1 | 1 | 0 | 1 | 1 | 1 | |
| Calpurnia | 0 | 1 | 0 | 0 | 0 | 0 | |
| Cleopatra | 1 | 0 | 0 | 0 | 0 | 0 | |
| mercy | 1 | 0 | 1 | 1 | 1 | 1 | |
| worser | 1 | 0 | 1 | 1 | 1 | 0 | |
| … | | | | | | | |

**Pay attention!**

TDM: rows are terms, cols are docs

DTM: rows are docs, cols are terms

# Commonly Used Terms in Text Mining

- **The Boolean Retrieval Model** – This is one of the basic models used to retrieve information from the term-document incidence matrix by
  - Uses term-document incidence matrix so it avoids linear search of all documents.
  - Uses binary operations which are the fastest possible.

Assignment Project Exam Help
https://powcoder.com

To answer the query "Brutus AND Caesar AND NOT Calpurnia," we take the vectors for "Brutus," "Caesar," and "Calpurnia," complement (negate) the binary vector for "Calpurnia," and then do a bitwise "AND":
(110100) AND (110111) AND (101111) = (100100)

Add WeChat powcoder

- **The Indexed Notation of a Matrix** – A record of only the position of ones. Indexed notation of a matrix is a more compact way of representing sparse matrices.

# Building an index

Building an index is done by sorting and grouping. There are four major steps:

- Collect the documents to be indexed.
- Tokenize the text, turning each document into a list of tokens.
- Do linguistic preprocessing, producing a list of normalized tokens, which are the indexing terms.
- Index the documents in which each term occurs. Your index should consist of a dictionary and postings.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

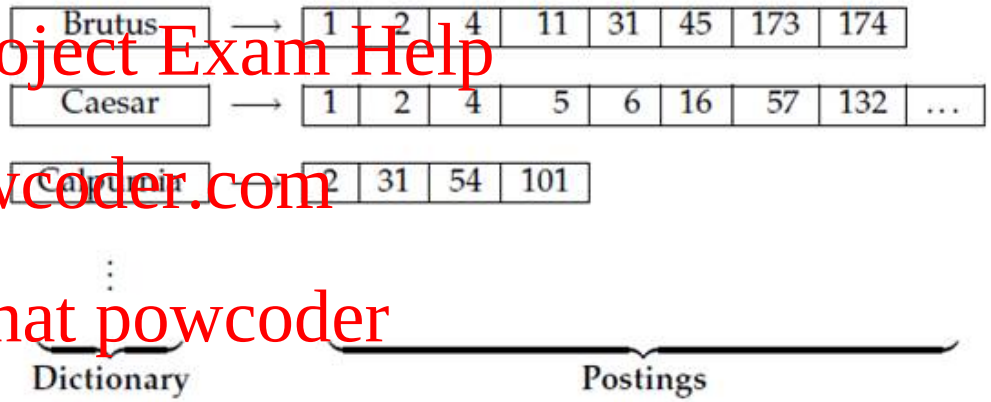
# Commonly Used Terms in Text Mining

- **Dictionary** (**Vocabulary** or **Lexicon**) – This keeps all the unique terms.
- **Postings** (**Posting Lists**) – Relates dictionary (unique) terms with document ID (the way we numbered documents).

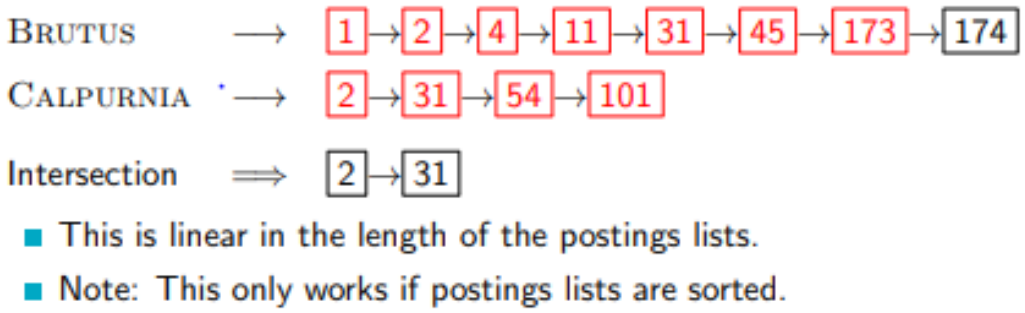
Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder




- Consider the query: Brutus AND Calpurnia




- This is linear in the length of the postings lists.
- Note: This only works if postings lists are sorted.

# Commonly Used Terms in Text Mining

- In Boolean queries, documents either match or don't
- Boolean queries often result in either too few or too many results
- It takes a lot of skill to come up with a query that produces a manageable number of hits. AND gives too few; OR gives too many
- **Ranked Retrieval Model** – Contrasting the Boolean model are ranked retrieval models such as the vector space model. These models use free text queries, and the system decides which documents best satisfy the queries.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

# Implementing Text Mining In R

1. Loading/Accessing Files (pdf, csv, txt, html, xml etc.).
2. Extracting textual context to electronic form (Create Corpus) – using **tm** package, specify tm
   - **readers** and
   - **sources**.
3. Preprocessing with **tm_map** - remove numbers, capitalization, common words, punctuation, and prepare your texts for analysis.
4. Staging the Data - create a document term matrix (**dtm**).
5. Explore your data - Organize terms by their frequency, export dtm to excel, clipboard etc.
6. Analysis
   - Analyze most frequent terms - Word Frequency
   - Plot Word Frequencies
   - Relationships Between Terms
   - Term Correlations
   - Word Clouds!
   - ML Analysis (Clustering by Term Similarity, Hierarchal Clustering, K-means clustering)

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

# Step 1. Loading/Accessing Files

- Extracting the content from a pdf file (OPTIONAL)
- Extracting the content from a text file

Assignment Project Exam Help

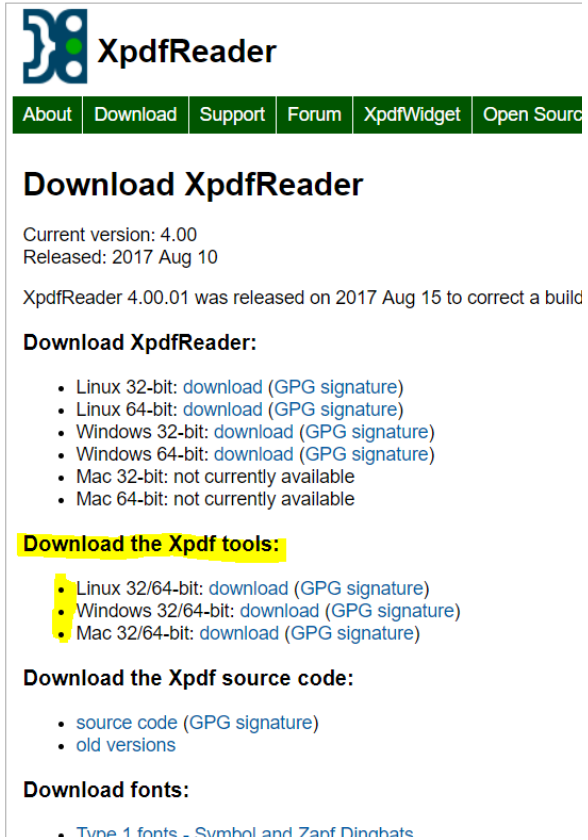
https://powcoder.com

Add WeChat powcoder

# Extracting the content from a pdf file (OPTIONAL)

- Installing Xpdf - Extracts the content from a pdf file
- You might find it helpful to use it in Text mining projects on your own.
- Install **pdftotext.exe** which is part of the Xpdf software suite (an open-source PDF viewer for the X Window System).
- It can be downloaded from: http://www.xpdfreader.com/download.html
- Choose the download for your operating system.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder



XpdfReader

About | Download | Support | Forum | XpdfWidget | Open Sourc

**Download XpdfReader**

Current version: 4.00
Released: 2017 Aug 10

XpdfReader 4.00.01 was released on 2017 Aug 15 to correct a build

**Download XpdfReader:**

- Linux 32-bit: download (GPG signature)
- Linux 64-bit: download (GPG signature)
- Windows 32-bit: download (GPG signature)
- Windows 64-bit: download (GPG signature)
- Mac 32-bit: not currently available
- Mac 64-bit: not currently available

**Download the Xpdf tools:**

- Linux 32/64-bit: download (GPG signature)
- Windows 32/64-bit: download (GPG signature)
- Mac 32/64-bit: download (GPG signature)

**Download the Xpdf source code:**

- source code (GPG signature)
- old versions

**Download fonts:**

- Type 1 fonts - Symbol and Zapf Dingbats

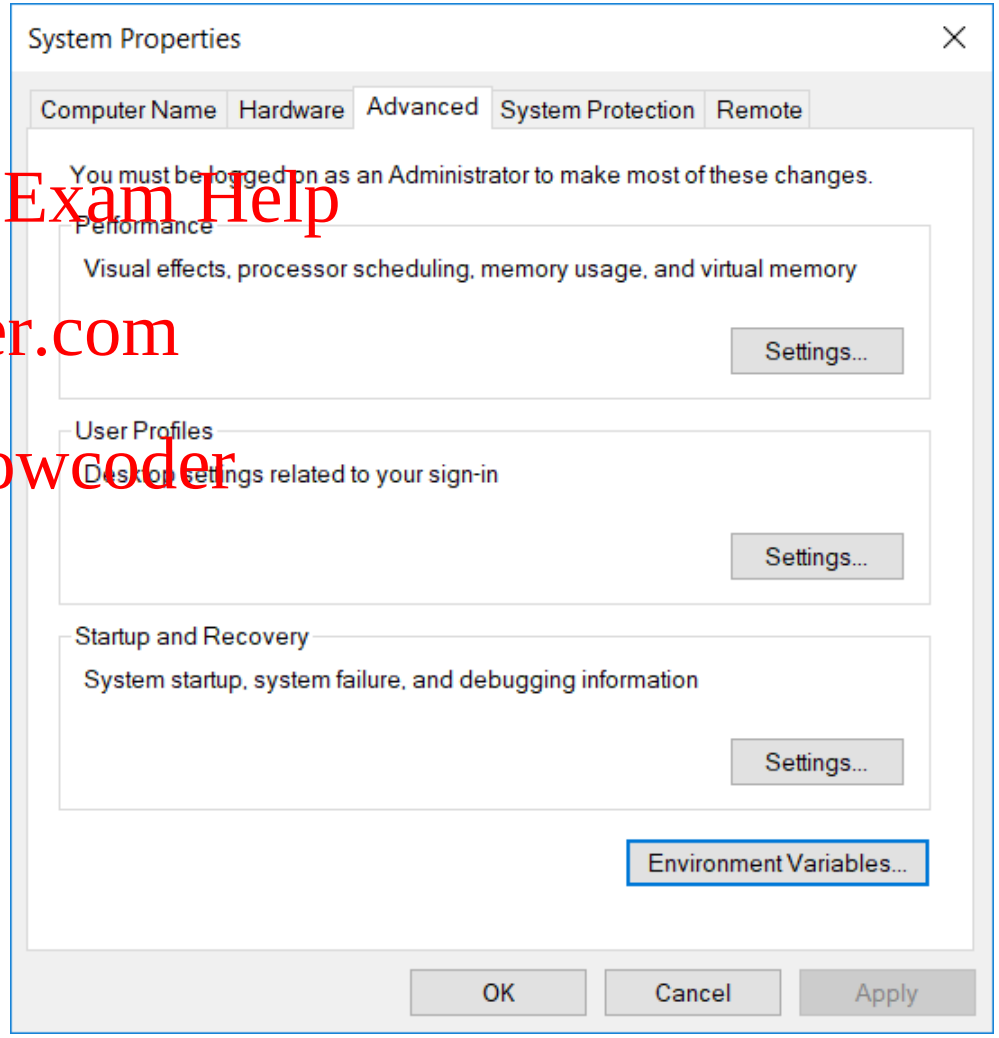
# Installing Xpdf (OPTIONAL)

- For Mac users you can use the link on the previous page or follow this one: http://macappstore.org/pdftotext/
  - Also see info in Blackboard module

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- Un-compress the downloaded files in a folder and set the environmental variables and the path to point to the folder where downloaded files are so R can find them.

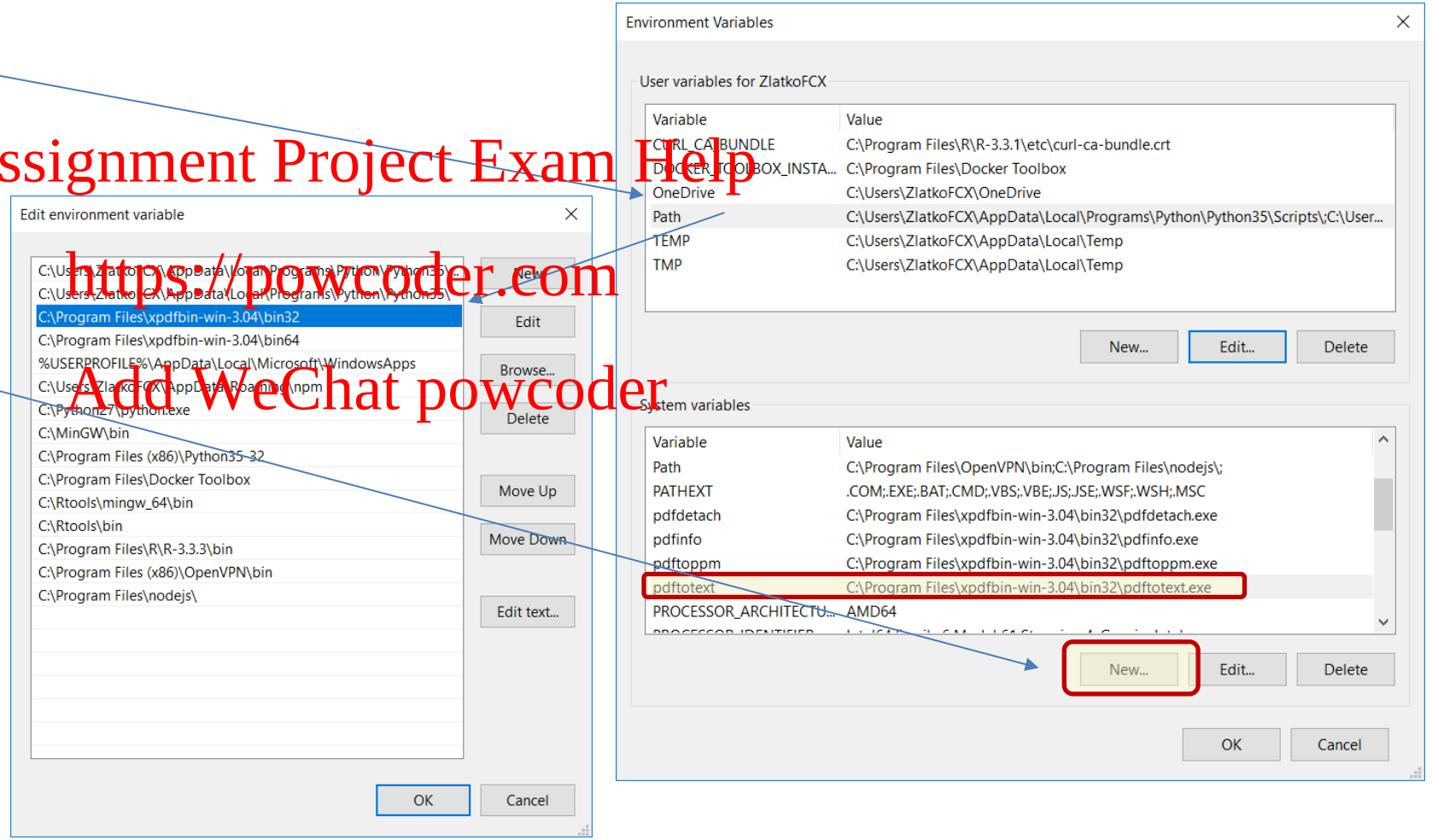
- On the Windows operating system you would do that by going to "System", choosing "Advanced system settings", and "Environment Variables".



# Installing Xpdf (OPTIONAL)

- Add to the “Path” user-variable the **folder** where the downloaded Xpdf files are.
- By clicking “New” add to the System the 2 new environmental variables with “Variable name” "pdfinfo" and "pdftotext".
- For the “Variable value” enter the **path** to the executables "pdfinfo.exe" and "pdftotext.exe".

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder



# Additional Xpdf Utilities (OPTIONAL)

These are all of the Xpdf utilities:

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- pdftotext -- generates a text file from a pdf file
- pdftops -- generates a PostScript file from a pdf file
- pdfinfo -- dumps a PDF file's Info dictionary (plus some other useful information)
- pdffonts -- lists the fonts used in a PDF file along with various information for each font
- pdfdetach -- lists or extracts embedded files (attachments) from a PDF (archived) file
- pdftoppm -- converts a PDF file to a series of PPM/PGM/PBM-format bitmaps
- pdfimages -- extracts the images from a PDF file

# Test the "pdftotext.exe" installation (OPTIONAL)

- To test the installation, convert a pdf file from a specific folder to a text file.

- The "pdftotext.exe" conversion from pdf to text in a terminal mode is implemented at as:

  > pdftotext "file.pdf"

Assignment Project Exam Help

https://powcoder.com

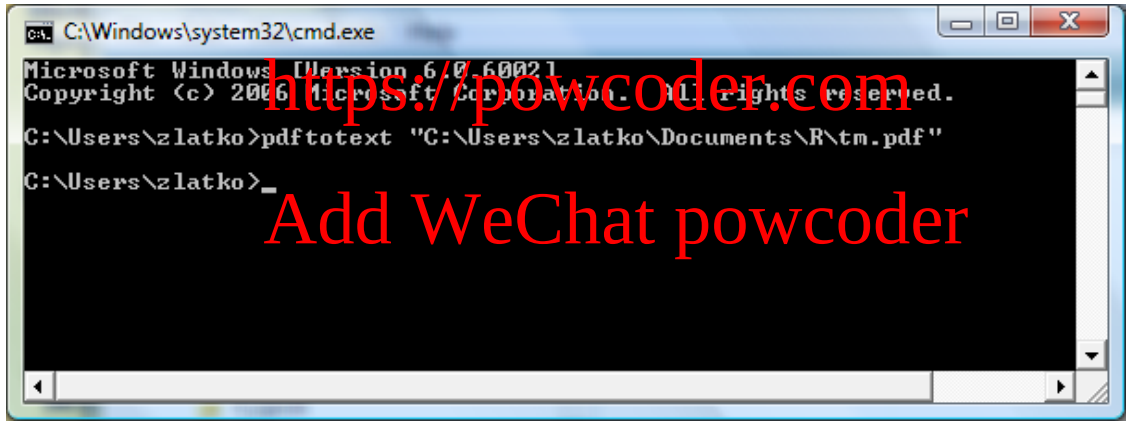
Add WeChat powcoder

- Note:
  - This usage produces a text file with the same name as the input file
  - The text file is created in the same directory as the PDFs.

# Test the "pdftotext.exe" installation (OPTIONAL)

- To test the installation, convert a pdf file from a specific folder to a text file.
- Open a terminal (cmd on Windows)
- Either navigate to the directory where the PDF file is or include the path to it in the filename.
- Note that the file name (and the path) needs to be enclosed in quotation.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder




- This usage produces a text file with the same name as the input file that is created in the same directory as the PDFs.
- In R we can use system() function to implement a cmd command.

# Try implementing these R code examples (OPTIONAL)

**Task:** Use R to implement terminal mode command such as

```
> pdftotext "file.pdf"
```

- To accomplish this from an R script you can use *system()*
- To insert the "" around the pdf filename we can uses *paste0()*. Here is the R code example:

```
system(paste(Sys.which("pdftotext"), paste0('"', myPDFfiles[1], '"')), wait=FALSE)
```

Assignment Project Exam Help
https://powcoder.com

**Task:** How to extend this to multiple files in a folder?

- A wildcards (*), for example *pdftotext "*.pdf"* , for converting multiple files, cannot be used because pdftotext expects only one file name.

Add WeChat powcoder

- Using R's *lapply()* several pdf files that are contained in a single folder (specified by the R object "myPDFfiles") can be converted with an in line function such as this,

```
> lapply(myPDFfiles, function(i) system(paste(Sys.which("pdftotext") , paste0('"', i, '"')), wait = FALSE))
```

- Note how each PDF file converted into a text file is indexed by *"i"*

# Getting several pdf files from a folder (OPTIONAL)

**Task:** How to get path to **several** pdf files that are contained in a **single** folder.

```
# Example 1: Convert to text single pdf files that is contained in a single folder.
exe.loc <- Sys.which("pdftotext")  # location of "pdftotext.exe"
pdf.loc=file.path(getwd(),"PDF Files")  # folder "PDF Files" with PDFs

# Get the path (character vector) of PDF file names
myPDFfiles <- normalizePath(list.files(path = pdf.loc, pattern = "pdf",  full.names = TRUE))

# Convert single pdf file to text by placing "" around the character vector of PDF file name
system(paste(exe.loc, paste0("", myPDFfiles[1], "")), wait=FALSE)
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Note:

- *Sys.which* gives the path to *pdftotext.exe* (the one you set it up during installation).
- *file.path(getwd(),"PDF Files")* Gets the current folder (*getwd()*) and forms a path.
- *normalizePath()* Converts the file paths to a canonical form for the platform.
- *list.files()* Lists the files in a Directory/Folder.
- *system()* Invoke a system command.
- myPDFfiles[1] Access the first element (path to the PDF).

# Implementing Text Mining In R

It is very unlikely that you would need to create from scratch any of the text mining tools and terms such as TDM-term document matrix.

Assignment Project Exam Help

All of the R text mining packages (such as ***tm***) are already capable of doing the necessary math and creating for example TDM.

https://powcoder.com

In order to use the *tm* package you need to load it by typing:

Add WeChat powcoder

```
> library(tm) # Load the Text Mining package
```

Follow the text mining steps in the following exercises.

# Using the tm Package

The main structure for managing documents in *tm* is called Corpus (Latin for body).

- It is electronically stored texts from which we would like to perform our analysis.
- Corpora (plural of Corpus) are R objects held fully in memory.
- It can represent a single document or a collection of text documents.

Assignment Project Exam Help

*tm* **package readers** (types of digital files):

```
> getReaders()
[1] "readDOC"    "readPDF"    "readPlain"    "readRCV1"
[5] "readRCV1asPlain"  "readReut21578XML"    "readReut21578XMLasPlain"
[8] "readTabular"   "readXML"
```

https://powcoder.com

Add WeChat powcoder

*tm* **sources** (to get the files from):

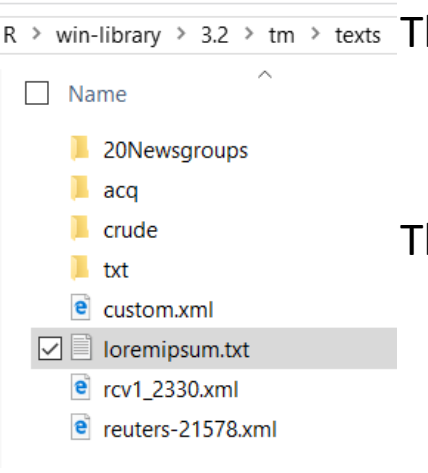
```
>  getSources()
[1] "DataframeSource" "DirSource"  "URISource"  "VectorSource"  "XMLSource"
```

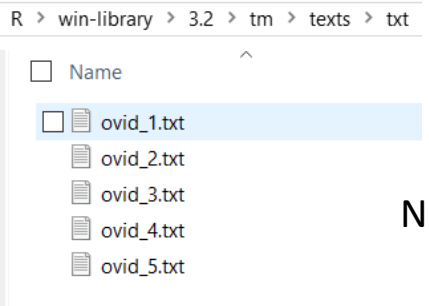
To avoid errors, make sure you use the appropriate source for your data.

- "DirSource" to get all the files in a folder, "URISource" for a specific file path.

# 2. Create Corpus - Content extraction from a Text File

Example of loading a sample of text documents and creating a corpus.




The *tm* package comes with 2 text files (2 poems in Latin)

- "loremipsum.txt"
- “txt/ovid_1.txt“

The following R (assuming *tm* is loaded) script extracts the content from these files into a corpus.

Assignment Project Exam Help

```
# Example R Script: Extracting text content from text files and load them as Corpus
loremipsum <- system.file("texts", "loremipsum.txt", package = "tm") # Path to "loremipsum.txt"
ovid <- system.file("texts", "txt", "ovid_1.txt", package = "tm") # Path to "ovid.txt"
Docs.pth <- URISource(sprintf("file://%s", c(loremipsum, ovid))) # Specify Source
corpus.txt<-VCorpus(Docs.pth) # load them as Corpus
inspect(corpus.txt)
```

https://powcoder.com

Add WeChat powcoder

Note: *system.file* creates the path in appropriate form for R on both Windows and Mac. *URISource* converts this to a syntax that *tm* prefers.

You can examine the first 5 lines of the "ovid.txt" corpus (second list element of corpus.txt) with

```
> corpus.txt[[2]]$content[1:5]
```

Optional: try to follow the lecture notes and extract the content from a PDF file.
(Not needed for class, you will be fine if you can handle TXT files.)

# Preparing for Assignment 3
# We will discuss much more on Thursday

Unzip the 20Newsgroups.zip file (Blackboard → Class Discussion → Live Classroom Slides) into your tm library.

This command will tell you where to extract the files:

```
> system.file("texts", package = "tm")
```

Go to this folder and unzip the 20Newsgroups.zip file there!

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

# Preparing for Assignment 3
# We will discuss much more on Thursday

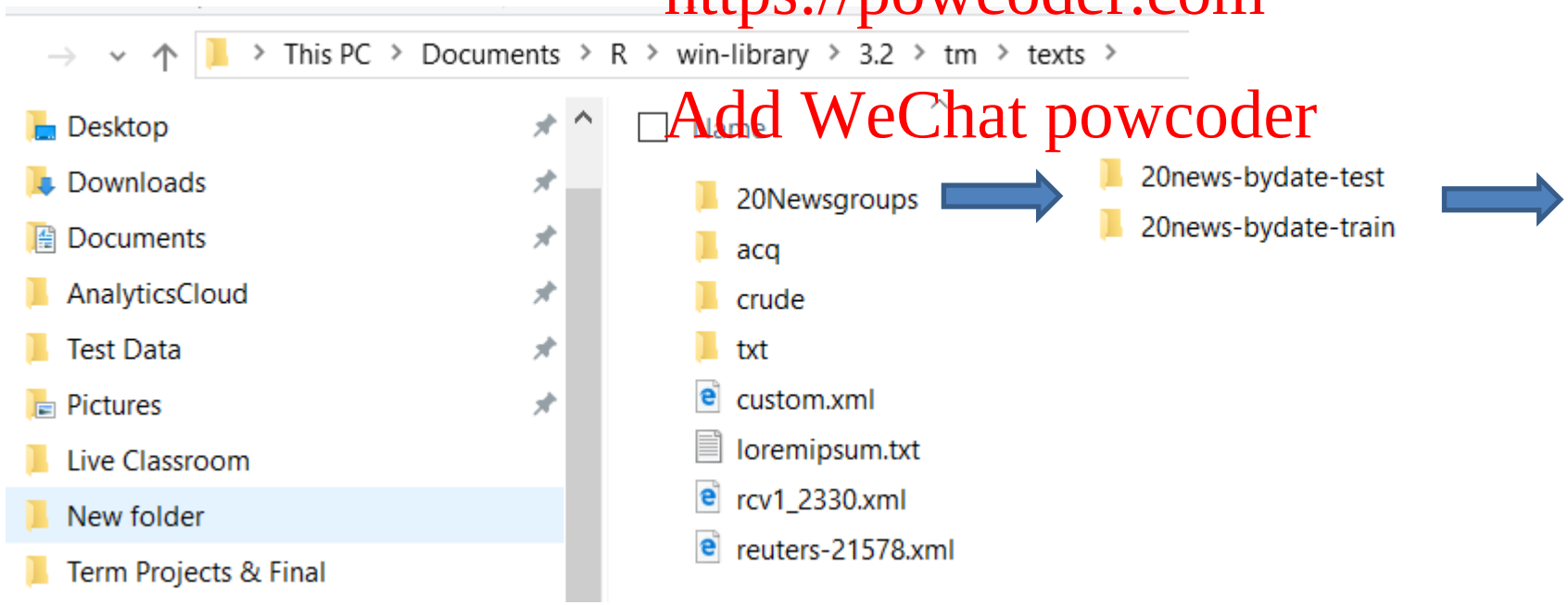
**Suggestion**: It is good to save the data in your "tm/text/" folder so you can specify path using **system.file()**

```
system.file("texts", "Folder Names to your files", package = "tm")
```

it easier to deal with the path in R.

Once you extract the files you will have 2 folders (with names "20news-bydate-test" and "20news-bydate-train") each containing 20 subfolders with files on different topics that you would need to perform text mining and analyze for similarity using R.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder



# •LIVE CLASSROOM 3 PART 2

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

# Step 3. Preprocessing – *tm* implementation

Once we have the corpora we need to perform some pre-processing transformations such as:

- converting the text to lower case
- removing numbers and punctuation
- removing stop words
- stemming and identifying synonyms

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- The basic transforms available within tm package can be seen with:

```
> getTransformations()
[1] "removeNumbers"   "removePunctuation" "removeWords"   "stemDocument"
[5] "stripWhitespace"
```

- These transformations are applied with the function "*tm_map()*".
- **Custom transformations** can be implemented by creating you own *R* function wrapped within tm's *content_transformer()* and then applying it to the corpus by passing it to *tm_map()*.

# Custom Transformation - Illustration

- Consider the following R script (*tm* loaded) that transforms the "@" into " ".

```
# Example: content_transformer()
email.texts <- c("Jon_Smith@bu.edu",
                 "Subject: Speaker's Info",
                 "I hope you enjoyed the lectures.",
                 "Here is address of the lecturer: ", "123 Main St. #25","Boston MA 02155")
# Step 1: Create corpus
y <- data.frame(doc_id = 1:length(email.texts), text = email.texts)
corpus.txt <- Corpus(DataframeSource(y))
# Step 2: "content transformer()" crate a function to achieve a custom transformation
transform.chr <- content_transformer(function(x, pattern) gsub(pattern, " ", x))
docs.tranf <- tm_map(corpus.txt, transform.chr, "@")
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- Note
  - The data is in a DataFrame format, so *tm*'s DataframeSource is used to create the corpus.
  - How the function "*transform.chr*" is created. It transforms the character that is passed to it.
  - > ? gsub will give you more info on R's standard pattern matching and replacement capabilities.
  - How the *content_transformer()* is implemented on the Corpus using *tm_map()* .
- For more examples see the lecture notes.

# 1. Transforms using *tm_map()*

Numbers may or may not be relevant to given analyses. This transform can remove numbers simply by

```
> docs.tranf <- tm_map(corpus.txt, removeNumbers)
```

In a similar fashion the punctuation can be removed by

```
> docs.tranf <- tm_map(corpus.txt, removePunctuation))
```

Assignment Project Exam Help

Stop words are common words found in a language. Words like "for"," very", "and", "of", "are", etc, are common stop words in the English language. We can list them (the ones provided by tm) with:

https://powcoder.com

```
> stopwords("english")
```

Add WeChat powcoder

They can be removed with

```
> docs.tranf <- tm_map(corpus.txt, removeWords, stopwords("english")))
```

In addition we can remove user defined stop words (such as " address " and "MA") with:

```
> docs <- tm_map(docs, removeWords, c("address", "MA"))
```

# 2. Transforms using *content_transformer()*

The first line of the corpus text contains an email address.

```
> corpus.txt[[1]]
> Jon_Smith@bu.edu
```

Assignment Project Exam Help

To implement multi character transformation such as "@" and "#" into a " ", we can pass them to "transform.chr" *(same function we made two slides back)* by separating them with logical OR "|", such as:

https://powcoder.com

```
> docs.tranf <- tm_map(corpus.txt, transform.chr, "@|#")
```

Add WeChat powcoder

The character "@" got transformed the into " "

```
> docs.tranf[[1]]
> Jon_Smith bu.edu
```

because the function "transform.chr" transforms the character that is passed to it into a space (" ").

We can see that this transformed lines 1 and 4 of email.texts object where the characters "@" and "#" appear.

# 3. Replace words

Specific Transformations

Sometimes we would like to perform a more specific transformations such as replacing words. This is illustrated in the code below.

Assignment Project Exam Help

```
# Example 4 : Replacing a word with another one
transform.words <- content_transformer(function(x, from, to) gsub(from, to, x))
docs.tranf <- tm_map(corpus.txt, transform.words, "Jon_Smith", "Jane_Doe")
```

https://powcoder.com

Add WeChat powcoder

# Step 4. Staging - Creating a Document Term Matrix

- A document term matrix is a matrix with
  - documents as the **rows**
  - terms as the **columns**
  - a **count** of the frequency of words as the cells of the matrix.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- In the "tm" package

  > DocumentTermMatrix(Docs.corpus)

  is used to create this matrix.
- To inspect the document term matrix use

  > inspect()
- The transpose of the DocumentTermMatrix() is created with

  > TermDocumentMatrix(Docs.corpus)

# Step 5. Explore your DTM data

Exploring the Document Term Matrix

- The term frequencies can be obtained as a vector by converting the document term matrix into a matrix and summing the column counts. Obtaining the most frequent terms in the "tm.pdf" is illustrated with the following script.

Assignment Project Exam Help

```
# Example 5: Creating a Document Term Matrix
Docs.pth <- system.file(file.path("doc", "tm.pdf"), package = "tm") # Path to tm.pdf
Docs.corpus <- Corpus(URISource(Docs.pth),readerControl = list(reader = readPDF(engine = "xpdf")))
dtm <- DocumentTermMatrix(Docs.corpus) # Document Term Matrix
freq <- colSums(as.matrix(dtm)) # Term frequencies
ord <- order(freq) # Ordering the frequencies
freq[tail(ord)] # Most frequent terms
```

https://powcoder.com

Add WeChat powcoder

- The output of this script lists the most frequent terms

```
> freq[tail(ord)]
metadata  corpus     and     for    text     the
      18      21      23      23      27      86
```

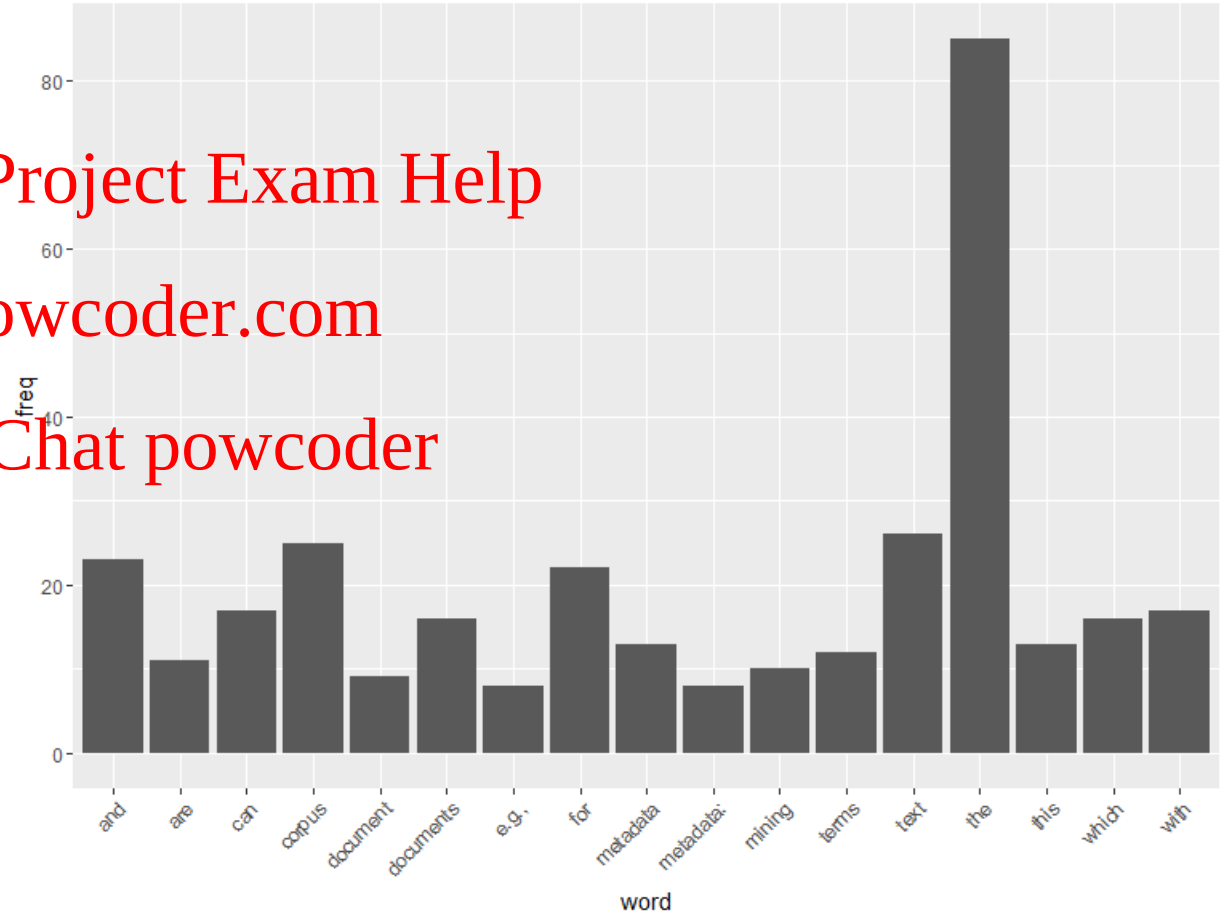
- For example the most frequent terms in the corpus that appear at least 10 times can be seen with

```
> findFreqTerms(dtm, lowfreq=10)
```

# Histogram of Word Frequencies

- In this graph:
  Words with freq > 7 are
  alphabetically ordered.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

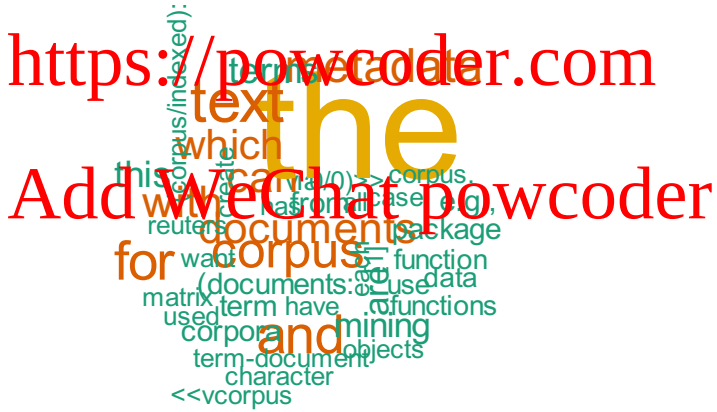


# Distribution of Term Frequencies

- For a quick visual overview of the frequency of words in a corpus we can generate a word cloud with:

```
> library(wordcloud)
> set.seed(123)
> wordcloud(names(freq), freq, min.freq=5, colors=brewer.pal(6, "Dark2"))
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder




- If needed the document term matrix can be converted to a simple matrix and saved.

```
> matdtm <- as.matrix(dtm)
> write.csv(matdtm, file=" dtm.csv")
```

# Step 6. Analysis - Tagging and Classification

- Classification - Supervised learning
  - We know the groups (tags) into which we separate data.
  - Tags are specific, relevant keywords by which we separate the text metadata.
  - The process of classification provide automated tagging.
  - The term "categorization" refers to assigning a category to an object.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- Training needed.
- So typically the available data is split into several datasets.
  - Training dataset (60%) – data for which we know the classification, used for training the algorithm.
  - Cross validation dataset (20%) – data on which we determine the statistical effectiveness of the classification.
  - Test dataset (20%) – data we analyze, to draw conclusions.

- **Note** in the next example, is good enough to split the data into only 2 datasets
  - Training dataset (50%)
  - Test dataset (50%)

# Classification Algorithms

- There are several classification algorithms.

- Here we will consider the k-NN algorithm. In the k-NN algorithm,
  - The term frequency, combined with inverse document frequency (TF-IDF), is used as the default weighting measure.
  - Cosine similarity (see math section of lecture notes) as the default similarity metric.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- The arguments of the function *knn()* are

  knn(train, test , cl, k)

  - **train** is a dataset for which classification is already known.
  - **test** is a dataset you are trying to classify.
  - **cl** is a factor of correct answers for the training dataset
  - **k** # number of neighbors considered.

- **Note** the function *knn()* expects same number of **train** and **test** datasets.

# Example using the k-NN algorithm

```
# Example: kNN Text Classification
library("tm")
Doc1 <- "I spent 10K on my car. Compared to the prices of most cars in their class it was cheap. It is a red car so I like it and it has a lot of space."
Doc2 <- "I checked the car prices and I could not find a red car for under 10K. So the price was good even though it had a hole. I heard that it belonged to a movie star."
Doc3 <-"I like the red color, so I would buy a red car even if the car's price is over 10K."
Doc4 <-"I don't like red cars. The insurance for red cars is higher regardless of the price and I would not spend more than 10K. I like black cars."
Doc5 <- "A red giant star can curve the space to form a black hole. In absence of stars the space is flat."
Doc6 <- "With exception of the stars the space is filled with blackness making the black holes even harder to see."
Doc7 <- "Our sun is a small star and it will not end as a black hole. It does not have enough mass to curve the space."
Doc8 <- "Very few stars will end as black holes but still the space contains large number of black holes."
doc <- c(Doc1,Doc2,Doc3,Doc4,Doc5,Doc6,Doc7,Doc8) # Merge all strings
corpus <- Corpus(VectorSource(doc))

# Preprocessing
corpus.temp <- tm_map(corpus, removePunctuation) # Remove Punctuation
corpus.temp <- tm_map(corpus.temp, stemDocument, language = "english")# Perform Stemming
dtm <- as.matrix(DocumentTermMatrix(corpus.temp)) # Document term matrix

# Text Classification
library(class) # Using kNN
train.doc <- dtm[c(1,2,5,6),] # Dataset for which classification is already known
test.doc <- dtm[c(3,4,7,8),] # Dataset you are trying to classify
Tags <- factor(c(rep("cars",2), rep("space",2))) # Tags - Correct answers for the training dataset
prob.test<- knn(train.doc, test.doc, Tags, k = 2, prob=TRUE) # k-number of neighbors considered
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

# Example using the k-NN algorithm

- There are 8 simple documents containing few sentences.
  - The first 4 documents (document ID from 1 to 4) mostly refer to subject related to cars.
  - The other 4 documents (document ID from 5 to 8) relate to space.
- There are some overlapping words between these 2 categories (space, star, red, black etc.).
- Note:
  - All documents are combined into a Corpus using R's *c()* function.

    ```
    doc <- c(Doc1,Doc2,Doc3,Doc4,Doc5,Doc6,Doc7,Doc8) # Merge all strings
    ```

  - Preprocessing is implemented before *dtm* document term matrix is created.
  - The package "*class*" is used for implementing kNN.

    ```
    library(class) # Using kNN
    ```

  - The data from *dtm* is equally split into test and train datasets (*train.doc* and *test.doc* ).

    ```
    train.doc <- dtm[c(1,2,5,6),] # Dataset for which classification is already known
    test.doc <- dtm[c(3,4,7,8),] # Dataset you are trying to classify
    ```

  - The correct answers for the training dataset - the tags (groups) are specified as factor data format. This is required by the *knn()* classification function. Note the use of R's *c()* and *rep()* functions!

    ```
    Tags <- factor(c(rep("cars",2), rep("space",2))) # Tags - Correct answers for the training dataset
    ```

  - Finally the classification is implemented, specifying that we have k=2 classes. The classification probability is returned for each test dataset.

    ```
    prob.test<- knn(train.doc, test.doc, Tags, k = 2, prob=TRUE) # k-number of neighbors considered
    ```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

# Analyzing the output of the knn()

The classification probability is returned for each test dataset.

Remember the test dataset contained:

1. doc3 related to cars.
2. doc4 related to cars.
3. doc7 related to space.
4. doc8 related to space.

The classification was correct.

```
> a <- 1:length(prob.test)
> b <- levels(prob.test)[prob.test]
> c <- attributes(prob.test)$prob
> result <- data.frame(Doc=a, Predict=b,Prob=c)
> result
  Doc Predict Prob
1   1    cars    1
2   2    cars    1
3   3   space    1
4   4   space    1
> sum(c)/length(Tags) # Overall probability
[1] 1
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Note that if you change Doc8 to "My car is priced well." and rerun the script, the Doc8 is now classified with the "cars" tag with 50% probability.

```
> a <- 1:length(prob.test)
> b <- levels(prob.test)[prob.test]
> c <- attributes(prob.test)$prob
> result <- data.frame(Doc=a, Predict=b,Prob=c)
> result
  Doc Predict Prob
1   1    cars  1.0
2   2    cars  1.0
3   3   space  1.0
4   4   space  0.5
> sum(c)/length(Tags) # Overall probability
[1] 0.875
>
> sum(prob.test==Tags)/length(Tags) # % Correct Classification
[1] 1
```

Why?

**You would need to change the training set and the Tags accordingly.**

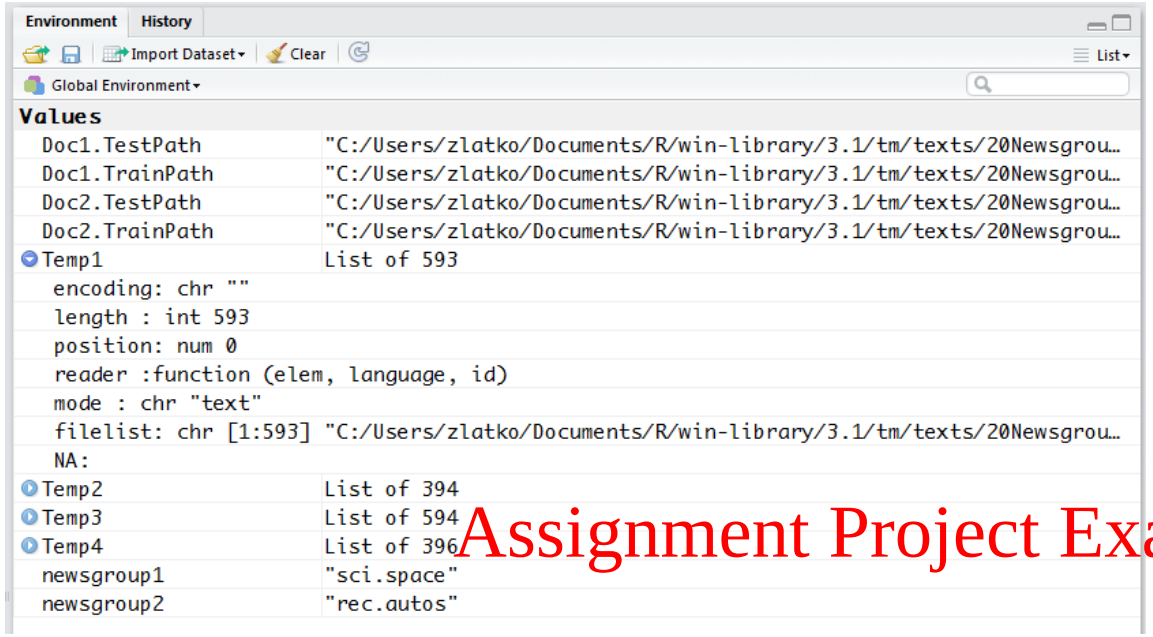
Finally the classification is not an exact process.

# Assignment 3 Hints

Get the Newsgroup data (**by date version data set**)

- Specify (the four) paths to saved Newsgroups data on your computer:
- Suggestion. Save data in your "tm/texts/" folder so you can specify path using **system.file()**
  - `system.file("texts", "20Newsgroups", package = "tm")`
- Specify *tm* source accordingly (for the 4 datasets) to create an R list of the paths
  - Note that you are going to import files from a folder, so you would need something like
  - `> Temp1 <- DirSource(dd1.TestPath)`
  - `Temp1$filelist[1]` gives "~/R/win-library/3.1/tm/texts/20Newsgroups/20news-bydate-test/sci.space/61242"
- Note that the size of the 4 loaded Newsgroups is different
  - sci.space Test is 394 files
  - sci.space Train is 593 files
  - rec.autos Test is 396 files
  - rec.autos Train is 594 files
- For each subject you need to select only 100 documents from Test and Train folders:
  - Many ways to do this, one is to just subset `Temp1$filelist[1:100]`

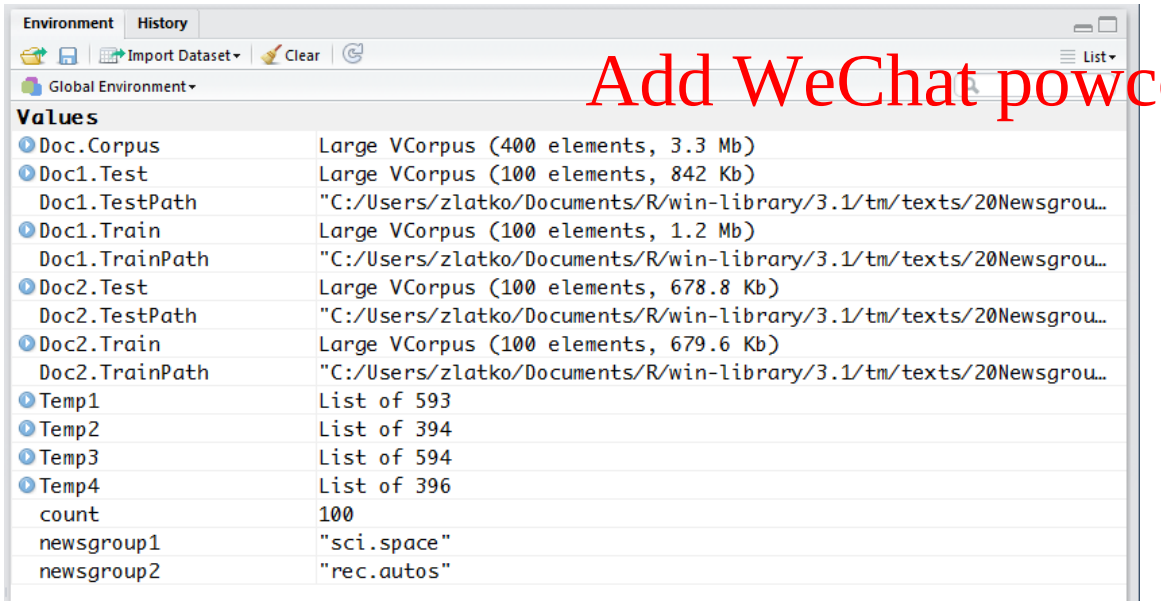
Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder




You can see the R objects you have created in the R Studio Workspace management Window

Note
-the paths to the data files
-The R list of the paths (Temp1)
-You can click on the blue dots to explore the R objects.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder




Note
-the Corpus properties, size and memory.
-You can explore them by click on the blue dots.

# Assignment 3 Hints

- Part A: Create the Corpus
  - `> Doc1.Train <- Corpus(URISource(Temp1$filelist[1:100]),readerControl=list(reader=readPlain))`

Assignment Project Exam Help

  - It may help to name your variables in a meaningful way as you load the different folders that you need:
    - Doc1.Train from the "sci.space" newsgroup train data
    - Doc1.Test from the "sci.space" newsgroup test data
    - Doc2.Train from the " rec.autos" newsgroup train data
    - Doc2.Test from the " rec.autos" newsgroup test data

https://powcoder.com

  - Merge all of 4 Corpora into 1 Corpus keeping the above specified order

Add WeChat powcoder

    - Using R's *c()* function for example – you can *c()* together Corpus arguments and the result will still be a Corpus
    - Why 1 Corpus and not 4? So we can implement all the pre processing steps at once and create one DTM.

- Part B: Implement some preprocessing (clearly indicate what you have used)
  - Note that the end classification result depends on this step.
  - Remember *tm* uses *tm_map()* for basic and custom transforms.
  - See examples in lecture notes.
  - Note that you can implement some of the preprocessing inside DocumentTermMatrix , see
    `> ?DocumentTermMatrix`

# Assignment 3 Hints

- Part C: The assignment asks you to enforce minimum word length and number of times that each word needs to appear (across the entire corpus, not on a per document basis).
- One way to accomplish this:
  - `DocumentTermMatrix(Doc.train.temp, control=list(wordLengths=c(2, Inf), bounds=list(global=c(5,Inf)) ))`

Assignment Project Exam Help

https://powcoder.com

Part D:

- First it asks you to split the Document-Term Matrix into
  - **Train** DTM containing only those rows of the DTM from Part C that pertain to **training** data for both sci.space and rec.autos
  - **Test** DTM containing only those rows of the DTM from Part C that pertain to **test** data for both sci.space and rec.autos

Add WeChat powcoder

- Then: Create a 200-length vector identifying which topic identifies each document in the **training** corpus.
  - Example: If the first document came from sci.space, then this vector should start with "sci" (or something else that you will remember means that the document came from sci.space)
  - Most likely you will have a lot of "sci" (how many??) and then a lot of "rec" (how many??)
- Classify text using the kNN() function
- Display Classification Results (it can be the same as in lecture examples).

# Assignment 3 Hints

The final Part D result should look something like this:

```
> result
    Doc Predict      Prob Correct
1     1     Sci 0.5000000    TRUE
2     2     Rec 0.5000000   FALSE
3     3     Sci 0.5000000    TRUE
4     4     Rec 0.5000000   FALSE
5     5     Sci 1.0000000    TRUE
6     6     Rec 0.5000000   FALSE
7     7     Sci 1.0000000    TRUE
8     8     Sci 0.5000000    TRUE
9     9     Rec 0.5000000   FALSE
10   10     Rec 0.5000000   FALSE
11   11     Sci 0.5000000    TRUE
12   12     Rec 0.5000000   FALSE
13   13     Rec 0.5000000   FALSE
14   14     Rec 1.0000000   FALSE
15   15     Sci 0.5000000    TRUE
16   16     Rec 0.5000000   FALSE
17   17     Rec 0.5000000   FALSE
18   18     Sci 1.0000000    TRUE
19   19     Sci 1.0000000    TRUE
20   20     Sci 0.5000000    TRUE
21   21     Sci 0.5000000    TRUE
22   22     Rec 0.5000000   FALSE
23   23     Rec 0.6666667   FALSE
```

Part E: Percentage of documents predicted correctly

- The last column should help with this!

Part F: does the Prob column in Part D relate to the answer to Part E? In what way / explain?

Part G: give your opinion and defend it according to your understanding of what you've learned.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder