FIT3152 Data analytics – Lecture 11

Text analytics

- Overview
- Processing text for analysis
- Creating a Term-Document Matrix
- Document similarity calculations

Text analytics in R

- Text processing
- Document clustering

Week-by-week

Week Starting	Lecture	Topic	Tutorial	A1	A2
2/3/21	1	Intro to Data Science, review of basic statistics using R	...		
9/3/21	2	Exploring data using graphics in R	T1		
16/3/21	3	Data manipulation in R	T2	Released	
23/3/21	4	Data Science methodologies, dirty/clean/tidy data, data manipulation	T3		
30/3/21	5	Network analysis	T4		
6/4/21		Mid-semester Break			
13/4/21	6	Regression modelling	T5		
20/4/21	7	Classification using decision trees	T6	Submitted	
27/4/21	8	Naïve Bayes, evaluating classifiers	T7		Released
4/5/21	9	Ensemble methods, artificial neural networks	T8		
11/5/21	10	Clustering	T9		
18/5/21	11	Text analysis	T10		Submitted
25/5/21	12	Review of course, Exam preparation	T11		

SETU

Student Evaluation of Teaching and Units (SETU) has opened for Semester 1.

- All students are encouraged to participate. Your feedback is very important.
- You will see a block in Moodle linking you to the survey.
- There are 100, $50 vouchers for students to win.
- The University will email students a weekly reminder with links to the survey.

Assignment 2

The objective of this assignment is to gain familiarity with classification models using R.

- You will be using a modified version some Kaggle competition data, to predict cloud cover in Australia.
- The data contains a number of meteorological observations as attributes, and the class attribute “CloudTomorrow”.
- Parts 1 – 7 will be familiar from tutorials.
- Parts 8 – 11 are a bit more challenging and will require some independent learning and initiative.

Assignment 2

FIT3152 Data analytics: Assignment 2

This assignment is worth 20% of your final marks in FIT3152.

Due:	Friday 21st May 2021 11:55pm GMT+10
How to submit:	Submit your written report as a single pdf with R code pasted in as machine-readable text as an appendix, or as an R Markup document that contains the R code with the discussion/text interleaved. Render this as an HTML file and print off as a pdf and submit. Whichever method you choose, you will submit a single pdf, and your R code will be machine readable text. Use the naming convention: Firstname.Lastname.studentID.pdf. Upload the file to Moodle. Do not zip. Do not submit your data file.

Objective:

The objective of this assignment is to gain familiarity with classification models using R.

You will be using a modified version of the Kaggle competition data: Predict next-day rain in Australia. https://www.kaggle.com/jsphyg/weather-dataset-rattle-package, but predicting whether or not the following day will be cloudy. The data contains a number of meteorological observations as attributes, and the class attribute "CloudTomorrow". Details of the decision attributes follow the assignment description.

You are expected to use R for your analysis, and may use any R package. Clear your workspace, set the number of significant digits to a sensible value, and use 'WAUS' as the default data frame name for the whole data set. Read your data into R using the following code:

Assignment 2

```
rm(list = ls())
WAUS <- read.csv("CloudPredict2021.csv")
L <- as.data.frame(c(1:49))
set.seed(88888888) # Your Student ID is the random seed
L <- L[sample(nrow(L), 10, replace = FALSE),] # sample 10 locations
WAUS <- WAUS[(WAUS$Location %in% L),]
WAUS <- WAUS[sample(nrow(WAUS), 2000, replace = FALSE),] # sample 2000 rows
```

We want to obtain a model that may be used to predict whether it is going to be cloudy tomorrow for 10 locations in Australia.

Assignment questions:

1. Explore the data: What is the proportion of cloudy days to clear days.? Obtain descriptions of the predictor (independent) variables – mean, standard deviations, etc. for real-valued attributes. Is there anything noteworthy in the data? Are there any attributes you need to consider omitting from your analysis? (1 Mark)

2. Document any pre-processing required to make the data set suitable for the model fitting that follows. (1 Mark)

3. Divide your data into a 70% training and 30% test set by adapting the following code (written for the iris data). Use your student ID as the random seed.

Assignment 2

```
set.seed(XXXXXXXX) #Student ID as random seed
train.row = sample(1:nrow(iris), 0.7*nrow(iris))
iris.train = iris[train.row,]
iris.test = iris[-train.row,]
```

4. Implement a classification model using each of the following techniques. For this question you may use each of the R functions at their default settings, or with minor adjustments to set factors etc. (5 Marks)

   • Decision Tree • Naïve Bayes • Bagging • Boosting • Random Forest

5. Using the test data, classify each of the test cases as 'cloudy tomorrow' or 'not cloudy tomorrow'. Create a confusion matrix and report the accuracy of each model. (1 Mark)

6. Using the test data, calculate the confidence of predicting 'cloudy tomorrow' for each case and construct an ROC curve for each classifier. You should be able to plot all the curves on the same axis. Use a different colour for each classifier. Calculate the AUC for each classifier. (1 Mark)

7. Create a table comparing the results in Parts 5 and 6 for all classifiers. Is there a single "best" classifier? (1 Mark)

8. Examining each of the models, determine the most important variables in predicting whether or not it will be cloudy tomorrow. Which variables could be omitted from the data with very little effect on performance? Give reasons. (2 Marks)

Assignment 2

9. Starting with one or some of the classifiers you created in Part 4, create a classifier that is simple enough for a person to be able to classify whether it will be cloudy or not tomorrow by hand. Describe your model, either with a diagram or written explanation. How well does your model perform, and how does it compare to those in Part 4? What factors were important in your decision and why you chose the attributes you used. (2 Marks)

10. Create the best tree-based classifier you can. You may do this by adjusting the parameters, and/or cross-validation of the basic models in Part 4, or using an alternative tree-based learning algorithm. Show that your model is better than the others using appropriate measures. Describe how you created your improved model, and why you chose that model. What factors were important in your decision and why you chose the attributes you used. (2 Marks)

11. Using the insights from your analysis so far, implement an Artificial Neural Network classifier and report its performance. Comment on attributes used and your data pre-processing required. How does this classifier compare with the others? Can you give any reasons? (2 Marks)

12. Write a brief report (suggested length 6 pages) summarizing your results in parts 1 – 10. Use commenting (# ----) in your R script, where appropriate, to help a reader understand your code. Alternatively combine working, comments and reporting in R Markdown. (2 Marks)

Assignment 2 Q&A

- Question 10 suggests using cross-validation for the ensemble classifiers, as one way of creating their best model. Some students were wondering how to do this, because they do not have examples in the tutorial.
  - > Correct, students need to research this themselves. I suggest using the package manuals as a starting point.
- In Tutorial 9 variable importance for Naive Bayes is not included in the solutions.
  - > Correct, NB doesn’t report variable importance.

Assignment 2 Q&A

- Rubric
  - > Q1 data exploration, data types, summary stats[1 mark]
  - > Q2 any pre-processing, how NA values handled [1 mark]
  - > Q4 implement each classifier [1 mark each]
  - > Q5 confusion matrix accuracy for each classifier [1 mark]
  - > Q6 plot ROC calculate AUC [1 mark]
  - > Q7 compare all classifiers, report best or no best [1 mark]
  - > Q8 variable importance [1] summary/analysis [1 mark]
  - > Q9i create simple model, report performance [1 mark]
  - > Q9ii describe factors in creation of model [1 mark]

Assignment 2 Q&A

- Rubric continued…
  - > Q10i create/describe single best classifier [1 mark]
  - > Q10ii comparison/discussion of why the best [1 mark]
  - > Q11i implement ANN [1 mark]
  - > Q11ii attributes used and pre-processing [1 mark]
  - > Q12i report writing of a high standard [1 mark]
  - > Q12ii quality of R script [1 mark]

End of semester exam

The end of semester exam:

- Will be online, e-vigilated. The university will advise you of the arrangements for sitting the exam.
- The exam is closed book. You can have two blank sheets of paper for working.
- You may use a calculator: graphing, scientific or CAS are permitted.
- The practice exam has been setup as a mock exam. It is a good indicator of length/complexity. Link:
  https://student-eassessment.monash.edu/mod/quiz/view.php?id=3863
- Solutions will be released in Week 12.

Quick revision from last week:

Question 1

Clustering is __________ learning?

A. supervised
B. unsupervised

Question 2

To k-Means cluster the data, the optimal k is:

A. 2

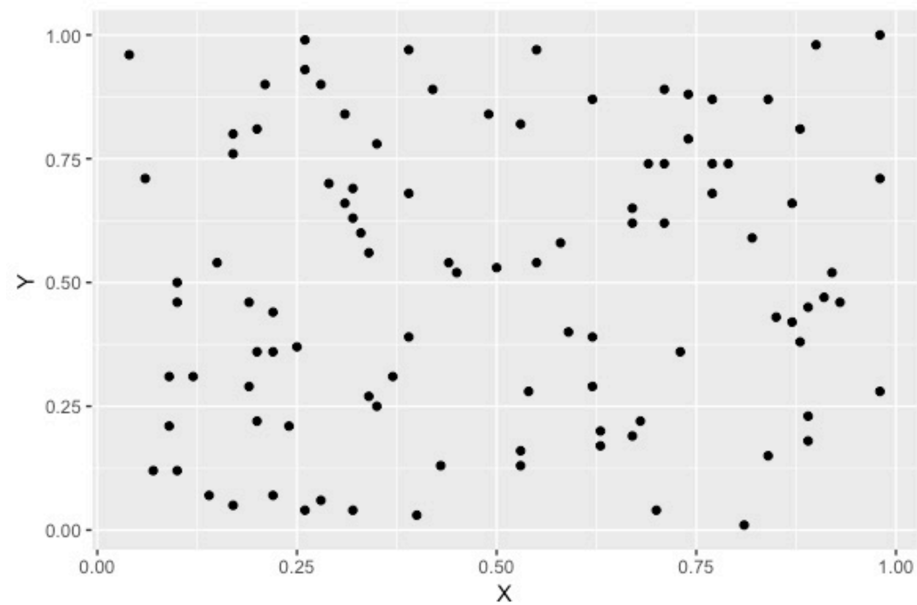
B. 4

C. 6

D. 8

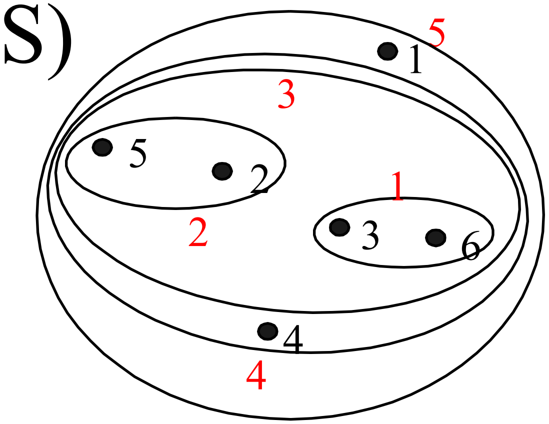
E. 10

F. Chosen by the user

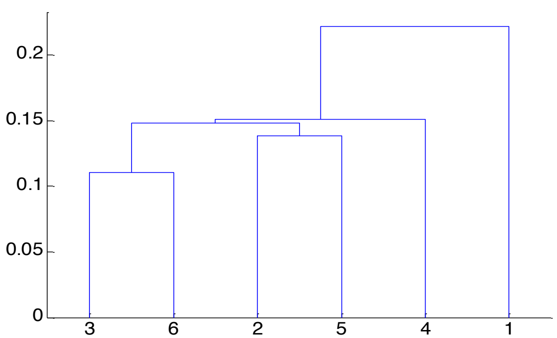


Question 3

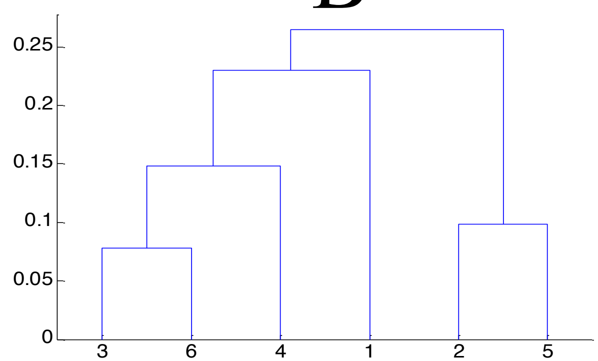
The enclosure diagram (RHS) corresponds to:



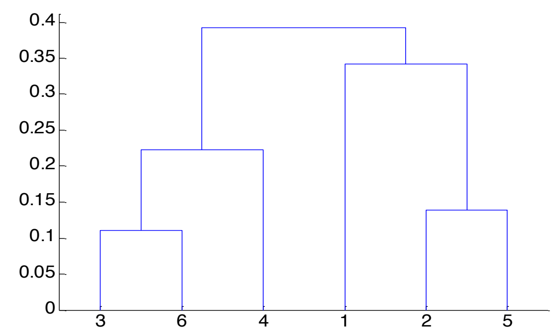
A



B



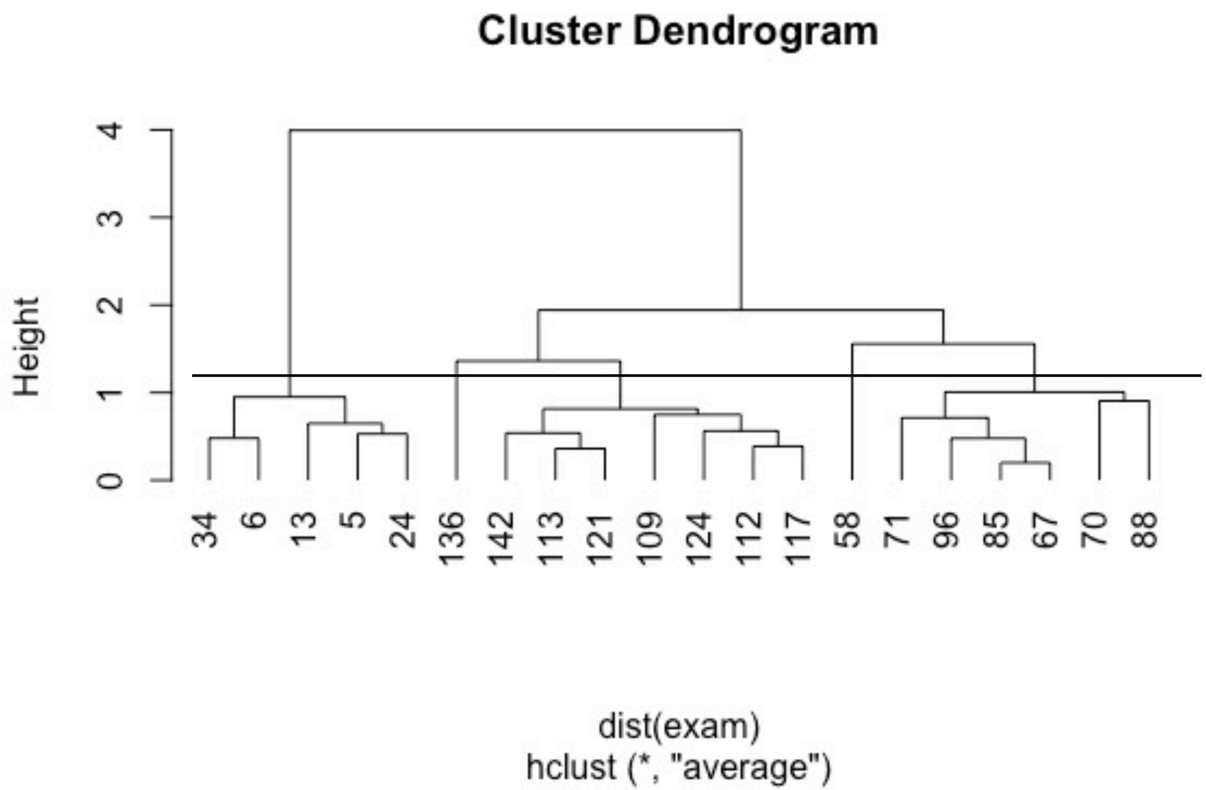
C



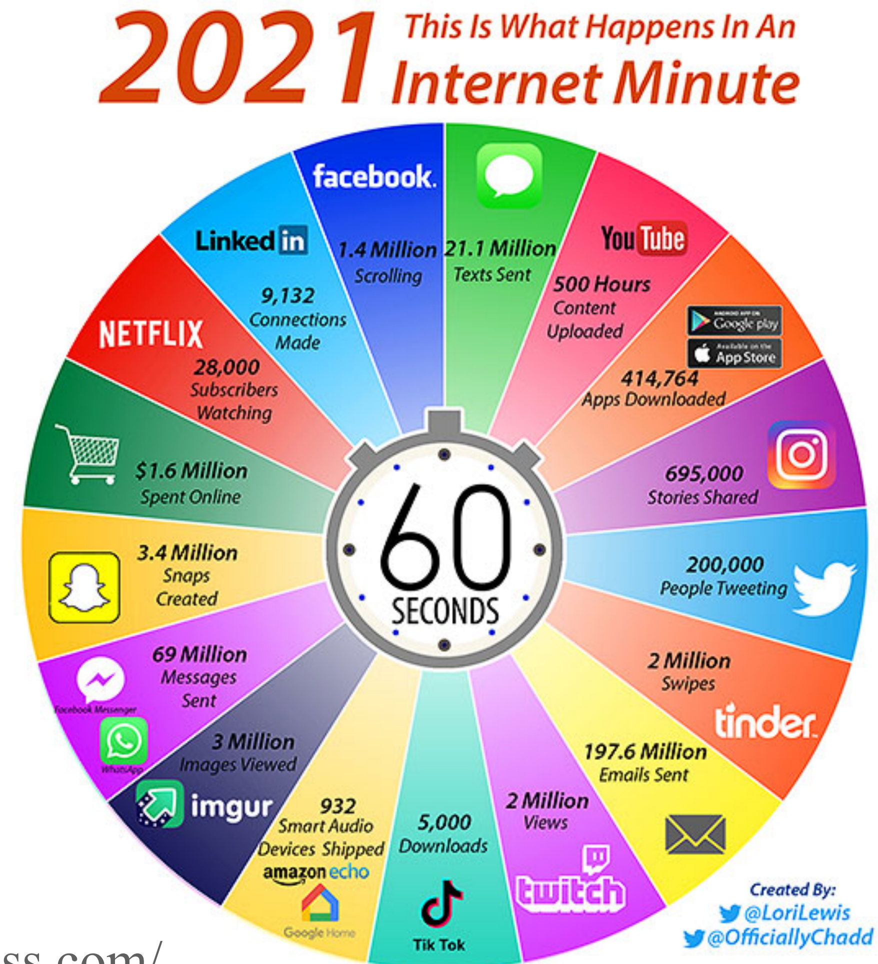
Question 4

This pruning divides the tree into ___ clusters:

A. 1
B. 2
C. 3
D. 4
E. 5
F. 6



Text: an important data source



https://www.allaccess.com/…

Text analytics:

Covers many different activities:

- Information retrieval
- Text mining (*traditional name and still popular*)
- Web mining
- Natural Language Processing (*words* + *semantics*)
- Document classification
- Document clustering
- Topic analysis

Text analytics:

Aims to extract useful knowledge from text data:

- Data is unstructured or semi-structured text
- Ultimate aim is to get meaning in an automated way
- Text data converted to counts, frequency distribution, correlations... (categories)
- Text data associated with sentiments via a library (for example LIWC as used in Assignment 1).

Sample data sources:

- Written documents, tweets, emails, news, web sites...

nytimes.com

How A.I. Steered Doctors Toward a Possible Coronavirus Treatment

By Cade Metz

In late January, researchers at BenevolentAI, an artificial intelligence start-up in central London, turned their attention to the coronavirus.

Within two days, using technologies that can scour scientific literature related to the virus, they pinpointed a possible treatment with speed that surprised both the company that makes the drug and many doctors who had spent years exploring its effect on other viruses.

https://www.nytimes.com/

nytimes.com

How A.I. Steered Doctors Toward a Possible Coronavirus Treatment

By Cade Metz

Over two days, a small team used the company's tools to plumb millions of scientific documents in search of information related to the virus. The tools relied on one of the newest developments in artificial intelligence — "universal language models" that can teach themselves to understand written and spoken language by analyzing thousands of old books, Wikipedia articles and other digital text.

https://www.nytimes.com/

Text analysis applications

Text classification applications

- Classifying emails to: filter out spam, sort into different folders
- Classifying document streams: identify news feeds of interest
- Sentiment analysis: determine public opinion

Text clustering

- Exploring text to find common words or features
- Discovering groups of documents with similar content

Text analysis – terminology

Document

- A piece of text (from a book to a single sentence), Tweet, Job application, Customer feedback, Emails, Blog posts, etc...

Term (or Token)

- Documents consist of individual token or terms – usually words

Corpus

- Collection of all documents to be analysed is called the corpus.

Feature set – Dictionary

- All features in the corpus (may be all words or terms, but often a reduced set of words or terms).

Text analysis overview

A commonly used approach to text analysis is the vector space model, where

- Document text is converted to a bag of words (or tokens).
- Counts of words are then treated as orthogonal vectors in n-dimensional space.
- The angle between documents indicates their degree of similarity.

Bag of words

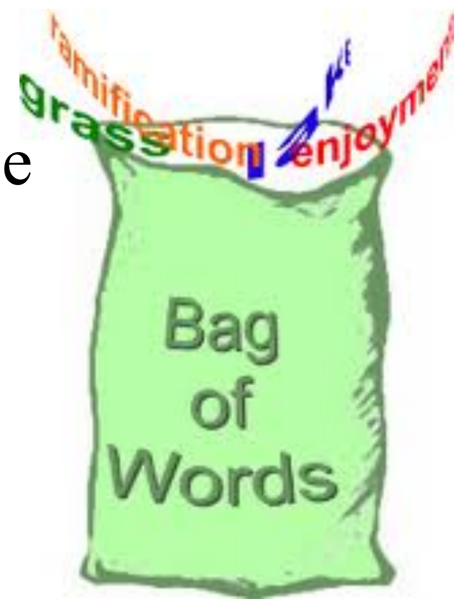
The vector space model uses a ‘bag of words’ approach.

- Each document assumed to be just a collection of words
- Makes implicit assumptions that the order of the words in a document does not matter
- Syntactically similar documents are semantically similar – which is often the case

These assumptions not always valid e.g.

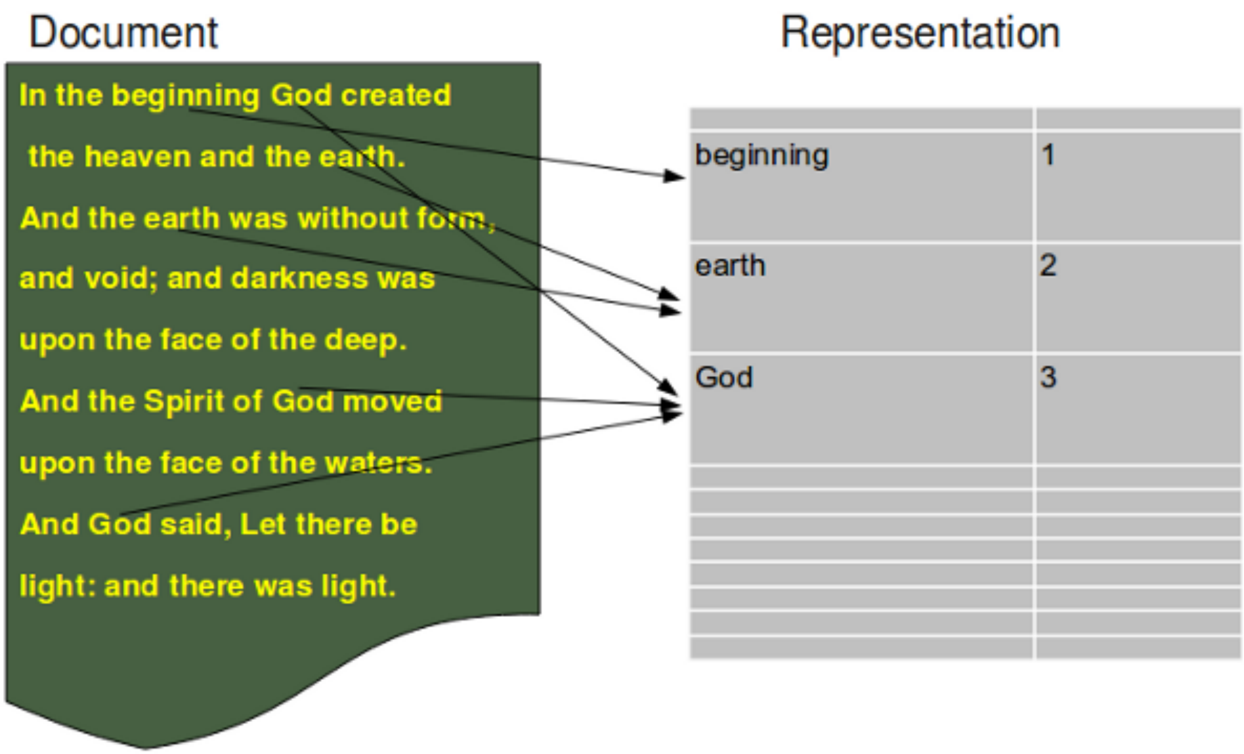
- ‘James and the giant ate a peach.’
- ‘The giant ate James and a peach.’

But works well in practice…



Term-Document Matrix

A Term-Document Matrix (TDM) represents each document by a vector of words:



Vector space model

Key words are extracted from all the documents

- A document is represented as a vector in high dimensional space corresponding to all the keywords
- Proximity of documents is measured using a similarity measure defined over the vector space

Issues:

- How to select keywords to capture "basic concepts" ?
- How to assign weights to each term?
- How to measure the similarity?

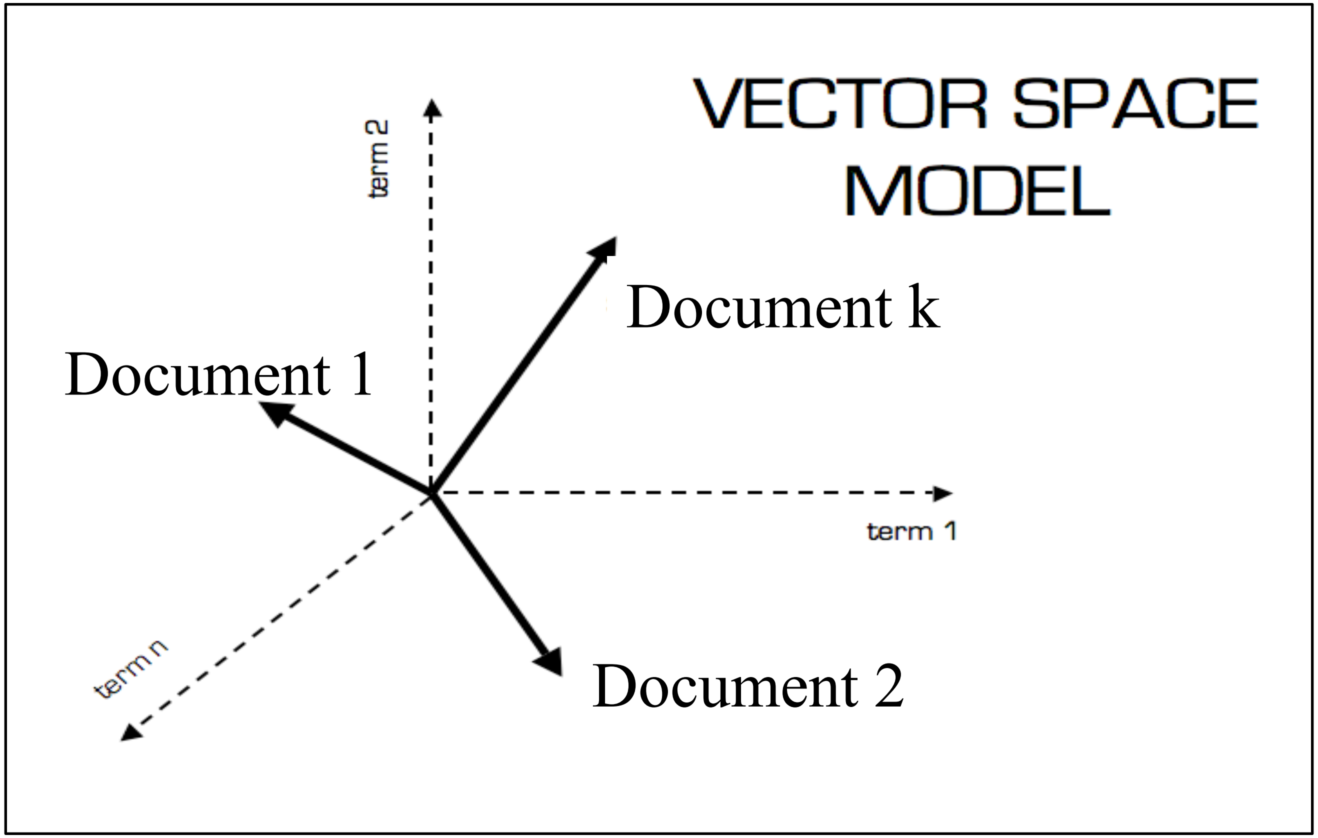
Vector space model

Term-Document Matrix for a corpus:

	T1	T2	T3	T4	T5	T6	T7	T8
Doc1	2	0	4	3	0	1	0	2
Doc2	0	2	4	0	2	3	0	0
Doc3	4	0	1	3	0	1	0	1
Doc4	0	1	0	2	0	0	1	0
Doc5	0	0	2	0	0	4	0	0
Doc6	1	1	0	2	0	1	1	3
Doc7	2	1	3	4	0	2	0	2

Vector space model

Representation of documents in the corpus:



Creating the Term-Document Matrix

A Term-Document Matrix (TDM) is created from all documents in the corpus:

- Doc1: {My dog ate my homework.}
- Doc2: {My cat ate the sandwich.}
- Doc3: {A dolphin ate the homework.}

Each column of the TDM represents a word (*or term*) and each row represents a document (*as a vector*).

Creating a vector from text

Corpus – 3 documents

- Doc1: {My dog ate my homework.}
- Doc2: {My cat ate the sandwich.}
- Doc3: {A dolphin ate the homework.}

Without further processing

- Tokens: a (1), ate (3), cat (1), dog (1), dolphin (1), homework (2), my (3), sandwich (1), the (2).

Term-Document Matrix - binary

The Corpus

Doc1: {My dog ate my homework.}
Doc2: {My cat ate the sandwich.}
Doc3: {A dolphin ate the homework.}

Binary Term Document matrix

	Terms								
Document	a	ate	cat	dolphin	dog	homework	my	sandwich	the
Doc 1	0	1	0	0	1	1	1	0	0
Doc 2	0	1	1	0	0	0	1	1	0
Doc 3	1	1	0	1	0	1	0	0	1

Term-Document Frequency - matrix

The Corpus

Doc1: {My dog ate my homework.}
Doc2: {My cat ate the sandwich.}
Doc3: {A dolphin ate the homework.}

Term Document Frequency matrix

	Terms								
Document	a	ate	cat	dolphin	dog	homework	my	sandwich	the
Doc 1	0	1	0	0	1	1	2	0	0
Doc 2	0	1	1	0	0	0	1	1	0
Doc 3	1	1	0	1	0	1	0	0	1

Refining the ‘bag of words’

The bag of words is improved by addressing the following:

- Upper and lower case words usually have the same meaning.
- Some frequently occurring words are not useful to discriminate between documents.
- Punctuation not useful.
- Tense may make similar words appear different.
- Groups of words may be important for meaning.

Extracting structure from text

Several steps:

- Tokenise
- Convert case
- Remove stop words
- Stem
- Lemmatize
- Create n-grams

Tokenisation

Tokenising breaks up the text into tokens.

- Text document is split into a stream of words;
- Remove all punctuation marks (“. ?, ! etc.);
- Replace tabs and other non-text characters by single white spaces;
- Merge all remaining words from all documents – this forms the dictionary of the documents collection (corpus).

Tokenisation: Example

Original:

- Doc1: {My dog ate my homework.}
- Doc2: {My cat ate the sandwich.}
- Doc3: {A dolphin ate the homework.}

Tokenised + case

- Doc1: {my | dog | ate | my | homework}
- Doc2: {my | cat | ate | the | sandwich}
- Doc3: {a | dolphin | ate | the | homework}

Tokens: a, ate, cat, dog, dolphin, homework, my, sandwich, the.

Filtering

Remove 'unnecessary' words from the dictionary, for example:

- Remove stop words – articles (a, an, the), conjunctions (and, but), prepositions (it, my, in, under).
- Remove commonly occurring words that may not assist with the clustering.
- Remove very infrequently occurring words.

Stop Words

Stop Words (also called noise words)

- Commonly occurring words such as: the, an, ...
- Words that are filtered out during the processing of the text, e.g.:
- Articles: a, am, the, of…
- Auxiliary verbs: is, are, was, were…

The process of filtering out these words is called Stopping.

There is no universal list of stop words – they are coded into the algorithm used.

Removing Stop Words: Example

Original:

- Doc1: {My dog ate my homework.}
- Doc2: {My cat ate the sandwich.}
- Doc3: {A dolphin ate the homework.}

Removing Stop Words

- Doc1: {dog | ate | homework}
- Doc2: {cat | ate | sandwich}
- Doc3: {dolphin | ate | homework}

Tokens: ate, cat, dog, dolphin, homework, sandwich.

Stemming

A stem is a natural group of words with the same (or very similar) meaning.

- Stemming reduces words to their stem or root form.
- Reduces the size of the dictionary.
- The Porter algorithm (1979) - most commonly used.

Examples:

- computational, computing... reduce to *comput*
- argue, argued, argues, and arguing reduce to *argu*

Stemming algorithms

Lovins, Paice, Snowball (used in R package tm),
Porter (most common for English 5 phases of word reduction)

- SSES → SS
  - **caresses → caress**
- IES → I
  - **ponies → poni**
- SS → SS
- S →
  - **cats → cat**
- EMENT →
  - **replacement → replac**
  - **cement → cement**

Example of different stemmers:
https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

Stemming: Example

Original:

- Doc1: {My dog ate my homework.}
- Doc2: {My cat ate the sandwich.}
- Doc3: {A dolphin ate the homework.}

Stemming (ate => eat) Since "ate" is the past tense of "eat".

- Doc1: {dog | eat | homework}
- Doc2: {cat | eat | sandwich}
- Doc3: {dolphin | eat | homework}

Tokens: eat, cat, dog, dolphin, homework, sandwich.

Lemmatization

More advanced form of stemming

- Takes context and 'part of speech' into account
- For example: 'meeting'

Stemming

- stems to meet

Lemmatization:

- Reduces to 'meet' when it is a verb
- Reduces to 'meeting' if it is a noun

Time consuming – stemming used more often.

Case normalisation

Converts the entire corpus to lower (or upper) case.

- In most cases an upper case version of a word should be treated no differently to the lower case version (e.g. upper case at the start of a sentence).
- Reduces the size of the dictionary (or feature set).

N-grams

Enhances the bag of words approach.

Frequent sequences of words are identified and included in the bag of words (in addition to the individual words).

N-grams: Example

Original:

- Doc1: {My dog ate my homework.}
- Doc2: {My cat ate the sandwich.}
- Doc3: {A dolphin ate the homework.}

Creating 2-grams

- Doc1: {dog | dog_eat | eat | eat_homework | homework}
- Doc2: {cat | cat_eat | eat | eat_sandwich | sandwich}
- Doc3: {dolphin | dolphin_eat | eat | eat_homework | homework}

TDM for processed documents

The Corpus:

- Doc1: {dog | dog_eat | eat | eat_homework | homework}
- Doc2: {cat | cat_eat | eat | eat_sandwich | sandwich}
- Doc3: {dolphin | dolphin_eat | eat | eat_homework | homework}

Term-Document (Frequency) Matrix

Document	eat	eat_homework	eat_sandwich	cat	cat_eat	dog	dog_eat	dolphin	dolphin_eat	homework	sandwich
Doc 1	1	1	0	0	0	1	1	0	0	1	0
Doc 2	1	0	1	1	1	0	0	0	0	0	1
Doc 3	1	1	0	0	0	0	0	1	1	1	0

Class Activity

Hand process a corpus

- From Pride and Prejudice, by Jane Austen:
- "You are over-scrupulous, surely. I dare say Mr. Bingley will be very glad to see you; and I will send a few lines by you to assure him of my hearty consent to his marrying whichever he chooses of the girls; though I must throw in a good word for my little Lizzy."

Tasks

- Tokenisation, • Filtering, • Removing stop words
- Case normalisation, • Stemming

Class Activity

- "You are over-scrupulous, surely. I dare say Mr. Bingley will be very glad to see you; and I will send a few lines by you to assure him of my hearty consent to his marrying whichever he chooses of the girls; though I must throw in a good word for my little Lizzy."

Class Activity (solution using R)

```
> writeLines(as.character(docs[[1]]))
"You are over scrupulous, surely. I dare say Mr.
Bingley will be very glad to see you; and I will
send a few lines by you to assure him of my
hearty consent to his marrying whichever he
chooses of the girls; though I must throw in a
good word for my little Lizzy."
> writeLines(as.character(docs[[1]]))
scrupul sure dare say mr bingley will glad see
will send line assur hearti consent marri
whichev choos girl though must throw good word
littl lizzi
```

Analysing text and documents

Term importance:

- Term Document Matrices
- Inverse Document Frequency

Document similarity

- Cosine Distance.

Term-Document Matrices

Term document frequency measures the frequency of a word for a specific document.

- Usually, very sparse, most entries = 0
- Terms should not be too common (not helpful for clustering). Arbitrary upper bound often placed on frequently occurring words
- Terms should not be too infrequent, those occurring very rarely often removed (not helpful for clustering)

Helpful to take into account overall frequency of a word in the entire corpus...

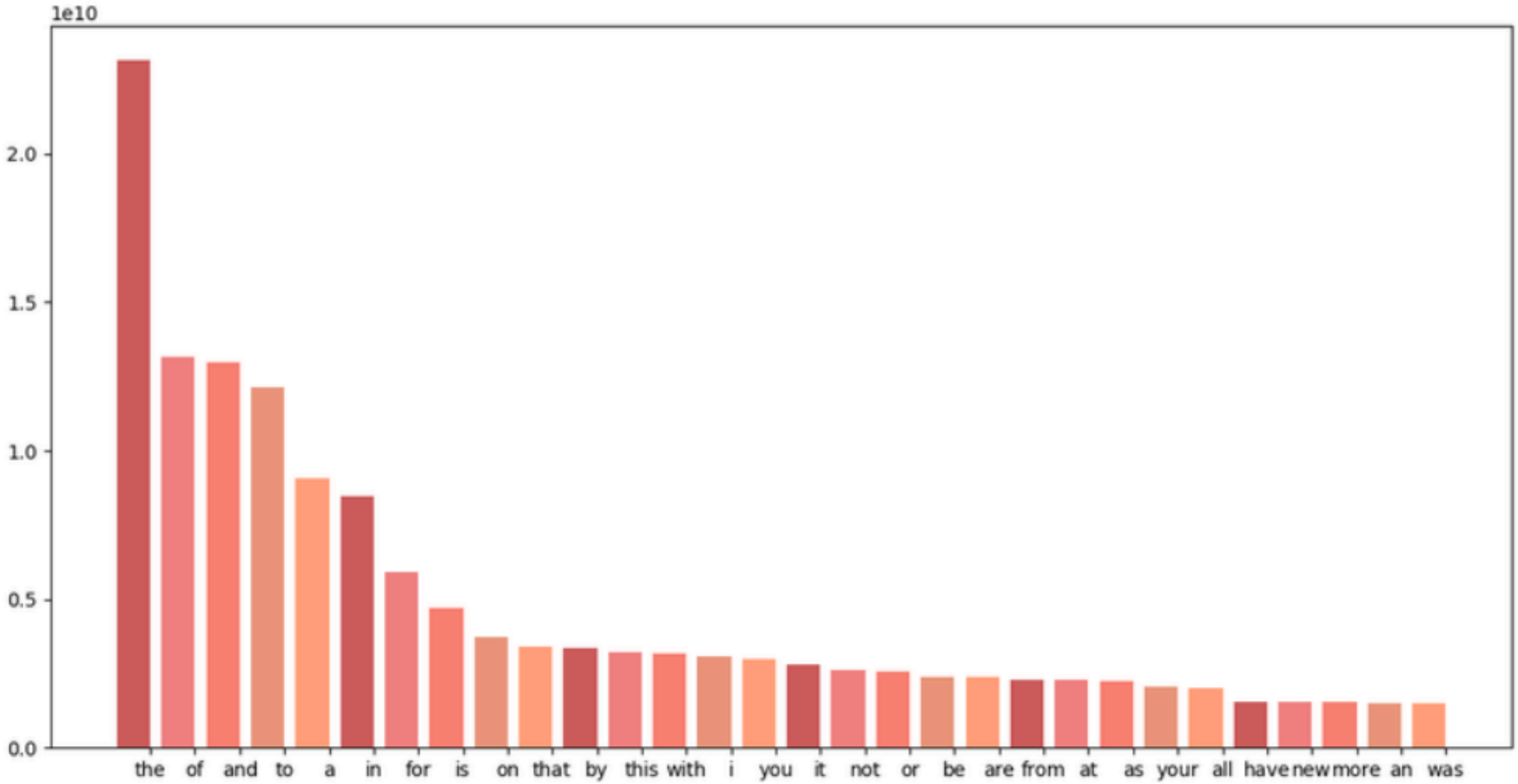
Inverse Document Frequency

Takes into account the relative number of documents in which a word occurs.

- Assumes that a word appearing in fewer documents is more likely to be important when it does occur.
- Gives a ‘boost’ to a term/word for being rare.
- If t is a term, then

$$\mathrm{IDF}(t) = 1 + \log\left(\frac{Total\ number\ of\ documents}{Number\ of\ documents\ containing\ t}\right)$$

English word frequency example



https://www.reddit.com/…

TF-IDF weighting

- TF(t,d) = the number of times term t appears in document d.
- TF = term frequency specific to one document
- IDF = inverse document frequency of a term in the entire corpus
- Terms can be weighted using a combination of TF and IDF
- Rationale – give a higher rating to less common terms in the documents that contain them multiple times

Combining TF and IDF

- TF and IDF are frequently multiplied to form TFIDF.
- Takes into account the frequency in a given document and the relative frequency of the term in the corpus.

$$TFIDF(t,d) = TF(t,d) \times IDF(t)$$

- Different definitions of TF and IDF exist – so software may not agree with manual calculations.

TFIDF Example

$TFIDF(t,d) = TF(t,d) \times IDF(t)$

Where: $\text{IDF}(t) = 1 + \log\left(\frac{Total\ number\ of\ documents}{Number\ of\ documents\ containing\ t}\right)$

and TF(*t*,*d*) = the number of times term *t* appears in document *d*.

For *t* = ‘at_homework’ in Document 1
TF(*t*,*d*) = 1
IDF(*t*) = 1+log(3/2) = 1.176
TFIDF = 1.176

Cosine distance similarity measure

Given two documents, made up of tokens:

$$D_i = (w_{i1}, w_{i2}, \cdots, w_{iN}) \qquad D_j = (w_{j1}, w_{j2}, \cdots, w_{jN})$$

The Cosine or normalised dot product:

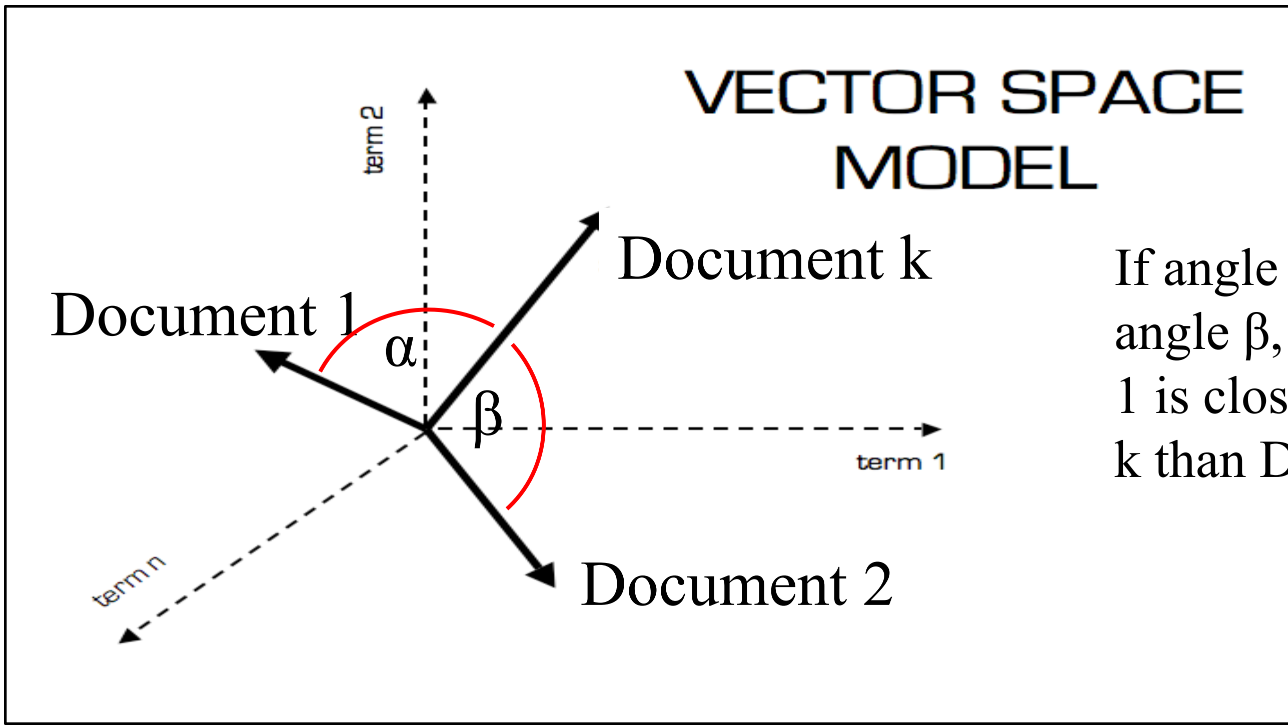
$$Cos(\theta) = Sim(D_i, D_j) = \frac{\sum_{t=1}^{N} w_{it} * w_{jt}}{\sqrt{\sum_{t=1}^{N} (w_{it})^2 * \sum_{t=1}^{N} (w_{jt})^2}}$$

gives a measure of the similarity between the documents in n-dimensional space, and is preferable to Euclidian distance for text! Can be used with unweighted (TDM) or weighted (TDFM, TFIDF) values.

Motivation behind cosine distance

Measures the angle between documents

Documents with smaller angle between them are closer when cosine distance is used



If angle α is less than angle β, then Document 1 is closer to Document k than Document 2 is

Cosine distance similarity measure

- Using TDM from the earlier example (with stop words removed, stemmed).

	Terms					
Document	**eat**	**cat**	**dog**	**dolphin**	**homework**	**sandwich**
Doc 1	1	0	1	0	1	0
Doc 2	1	1	0	0	0	1
Doc 3	1	0	0	1	1	0

$$Sim(D_1, D_2) = \frac{1*1+0*1+1*0+0*0+1*0+0*1}{\sqrt{3*3}} = \frac{1}{\sqrt{9}} = 0.333 = 70.5°$$

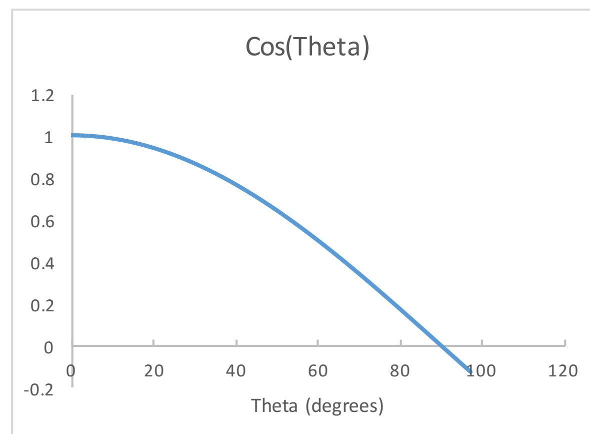
$$Sim(D_1, D_3) = \frac{1*1+0*0+1*0+0*1+1*1+0*0}{\sqrt{3*3}} = \frac{2}{\sqrt{9}} = 0.667 = 48.2°$$

$$Sim(D_2, D_3) = \frac{1*1+1*0+0*0+0*1+0*1+1*0}{\sqrt{3*3}} = \frac{1}{\sqrt{9}} = 0.333 = 70.5°$$

Cosine distance similarity measure

Interpreting cosine similarity:

- Closeness evaluated in n-dimensional space, where *n* = number of tokens in DTM.
- Angle between words is $Cos^{-1}\left(Sim(D_1, D_2)\right)$.
- Cosine similarity closer to 1 means documents are more similar than smaller values.
- Recall cosine function:



Text analysis

The term document matrix for a corpus of documents can be used for:

Classification – e.g. classifying documents into predefined classes

- Decision trees
- Naïve Bayes
- etc.

Clustering – to find documents with 'similar' content

- k-Means
- Agglomerative hierarchical clustering

Document clustering

Problem:

- Large volume of textual data. Billions of documents must be handled in an efficient manner. No clear idea of what documents are relevant for a given purpose.

Solution:

- Use document clustering (unsupervised learning).

Most popular document clustering methods are:

- k-Means clustering.
- Agglomerative hierarchical clustering.

Text analytics

Challenges:

- Very large dictionary.
- The size of actual documents relatively very small – e.g. tweets, abstracts (compared with dictionary).
- Correlation between words in a document.
- Varying sizes of documents – requires appropriate normalisation.

Text clustering steps

- Tokenisation
- Filtering, Removing stop words
- Case normalisation
- Stemming
- Lemmatization
- TF-IDF weighting
- Apply clustering algorithm

Text clustering in R

Install packages: tm, slam, SnowballC

Create a corpus: *Journal article abstracts in this case.*

TDM creation:

- Tokenisation, punctuation, stopping, stemming, case normalisation.

Text analysis, Removal of sparse terms.

Clustering.

References: download and use!

These were used to prepare the following slides:

Williams, G.

- Hands-On Data Science with R: Text Mining
  https://www.academia.edu/26073018/Hands_On_Data_Science_with_R…
- Introduction to the tm Package: Text Mining in R
  https://cran.**r**-project.org/web/**packages**/**tm**/vignettes/**tm**.pdf

R: Setup

```
> rm(list = ls())
> install.packages("tm") # requires R 3.3.1 or later
> install.packages("slam")
> install.packages("SnowballC")
> library(slam)
> library(tm)
> library(SnowballC) # Needed for stemming
```

R: Create corpus

```
> # folder named "CorpusAbstracts"
> # subfolder named "txt"
> cname = file.path(".", "CorpusAbstracts", "txt")
> cname
[1] "./CorpusAbstracts/txt"
> dir(cname)
[1] "Brachy01.txt" "Brachy02.txt" "DSR01.txt"
[4] "DSR02.txt"    "DSR03.txt"    "DSR04.txt"
[7] "IJOR.txt"     "IJPE.txt"     "IMA.txt"
[10] "JORS.txt"     "MS.txt"        "OptHed.txt"
[13] "StochI01.txt" "StochI02.txt" "StochI03.txt"
```

R: Create corpus

```
> docs = Corpus(DirSource((cname)))
> summary(docs)
              Length Class            Mode
Brachy01.txt 2      PlainTextDocument list
Brachy02.txt 2      PlainTextDocument list
DSR01.txt    2      PlainTextDocument list
DSR02.txt    2      PlainTextDocument list
DSR03.txt    2      PlainTextDocument list
DSR04.txt    2      PlainTextDocument list
IJOR.txt     2      PlainTextDocument list
IJPE.txt     2      PlainTextDocument list
...
```

R: Specific text transformations

```
> # Two functions to perform specific transformations
> # Key word to acronyn, ref Williams
> toJIT <- content_transformer(function(x, pattern)
  gsub(pattern, "JIT", x))
> docs <- tm_map(docs, toJIT, "Just-In-Time")
> # Hyphen to space, ref Williams
> toSpace <- content_transformer(function(x, pattern)
  gsub(pattern, " ", x))
> docs <- tm_map(docs, toSpace, "-")
```

R: Tokenization, stemming

```
> docs <- tm_map(docs, removeNumbers)
> docs <- tm_map(docs, removePunctuation)
> docs <- tm_map(docs, content_transformer(tolower))
> docs <- tm_map(docs, removeWords, stopwords("english"))
> docs <- tm_map(docs, stripWhitespace)
> docs <- tm_map(docs, stemDocument, language = "english")
```

Text: original

```
> writeLines(as.character(docs[[7]]))
```

```
This paper describes two decision tools that
allow identification of candidate components for
cost-effective Just-In-Time (JIT) replenishment.
The first is a simple and easily interpreted
coefficient, based on component cost and demand
parameters, which ranks the importance of
adoption of JIT replenishment of components. The
second is a procedure that can be used by
inventory managers to work from the ranking
coefficient to an approximate model for profit
and return on investment, for a given level of
inventory capitalisation, when the highest
priority JIT decisions are implemented...
```

Text: convert specific text

```
> writeLines(as.character(docs[[7]]))
This paper describes two decision tools that
allow identification of candidate components for
cost effective JIT (JIT) replenishment. The
first is a simple and easily interpreted
coefficient, based on component cost and demand
parameters, which ranks the importance of
adoption of JIT replenishment of components. The
second is a procedure that can be used by
inventory managers to work from the ranking
coefficient to an approximate model for profit
and return on investment, for a given level of
inventory capitalisation, when the highest
priority JIT decisions are implemented...
```

Text: numbers, punctuation, case

```
> writeLines(as.character(docs[[7]]))
this paper describes two decision tools that
allow identification of candidate components for
cost effective jit jit replenishment the first
is a simple and easily interpreted coefficient
based on component cost and demand parameters
which ranks the importance of adoption of jit
replenishment of components the second is a
procedure that can be used by inventory managers
to work from the ranking coefficient to an
approximate model for profit and return on
investment for a given level of inventory
capitalisation when the highest priority jit
decisions are implemented cumulatively we...
```

Text: stop words, white space

```
> writeLines(as.character(docs[[7]]))
paper describes two decision tools allow
identification candidate components cost
effective jit jit replenishment first simple
easily interpreted coefficient based component
cost demand parameters ranks importance adoption
jit replenishment components second procedure
can used inventory managers work ranking
coefficient approximate model profit return
investment given level inventory capitalisation
highest priority jit decisions implemented
cumulatively illustrate use tools case study
```

Text: stemming

> writeLines(as.character(docs[[7]]))

```
paper describ two decis tool allow identif
candid compon cost effect jit jit replenish
first simpl easili interpret coeffici base
compon cost demand paramet rank import adopt jit
replenish compon second procedur can use
inventori manag work rank coeffici approxim
model profit return invest given level inventori
capitalis highest prioriti jit decis implement
cumul illustr use tool case studi
```

R: Create Term-Document Matrix

```
> tdm <- DocumentTermMatrix(docs)
> #Inspect
> inspect(tdm[1:15, 1:4])
<<DocumentTermMatrix (documents: 15, terms: 4)>>
Non-/sparse entries: 5/55
Sparsity           : 92%
Maximal term length: 8
Weighting          : term frequency  (tf)
```

R: Create Term-Document Matrix

TermsDocs	absolut	abstract	accur	accuraci	
Brachy01.txt	0	0	0	0	
Brachy02.txt	0	0	0	0	
DSR01.txt	0	0	0	0	
DSR02.txt	0	0	0	0	
DSR03.txt	0	0	0	0	
DSR04.txt	0	0	0	0	
IJOR.txt	0	0	0	0	
IJPE.txt	1	0	0	0	
IMA.txt	0	0	0	0	
JORS.txt	0	0	0	0	
MS.txt	0	1	0	0	
OptHed.txt	0	0	0	0	
StochI01.txt	0	0	0	2	
StochI02.txt	0	0	1	0	...

R: Term frequencies

```
> # Word frequencies, ref Williams
> freq <- colSums(as.matrix(tdm))
> length(freq)
[1] 489
> ord = order(freq)
> freq[head(ord)]
absolut  abstract    accur   address algorithm
      1         1        1         1         1
> freq[tail(ord)]
cost        use      model inventori   queri
  23         25         30        32      33
```

R: Term frequencies

```
> # Frequency of frequencies, ref Williams
> head(table(freq), 10)
Freq
  1   2   3   4   5   6   7   8   9  10
223  94  45  35  22  13   6  11   8   6
> tail(table(freq), 10)
Freq
14 15 16 17 20 23 25 30 32 33
 2  2  2  1  3  1  1  1  1  1
```

R: Remove sparse terms

```
> dim(tdm) # Size of original Term Document Matrix
[1]  15 489
> dtms <- removeSparseTerms(tdm, 0.6) # rem. 60% empty
> dim(tdms)
[1] 15 13
> inspect(tdms)
<<DocumentTermMatrix (documents: 15, terms: 13)>>
Non-/sparse entries: 107/88
Sparsity           : 45%
Maximal term length: 9
Weighting          : term frequency (tf)
```

R: Remove sparse terms

Docs	base	can	cost	effici	high	...
Brachy01.txt	0	1	0	0	1	...
Brachy02.txt	3	0	0	0	2	...
DSR01.txt	0	0	4	0	1	...
DSR02.txt	0	0	1	1	0	...
DSR03.txt	1	0	3	0	0	...
DSR04.txt	0	0	4	2	1	...
IJOR.txt	1	1	2	0	0	...
IJPE.txt	0	2	2	1	0	...
IMA.txt	0	0	0	0	0	...
JORS.txt	0	1	2	1	0	...
MS.txt	0	0	0	0	0	...
OptHed.txt	1	0	2	0	0	...
StochI01.txt	1	1	0	2	1	...
StochI02.txt	2	1	0	2	1	...

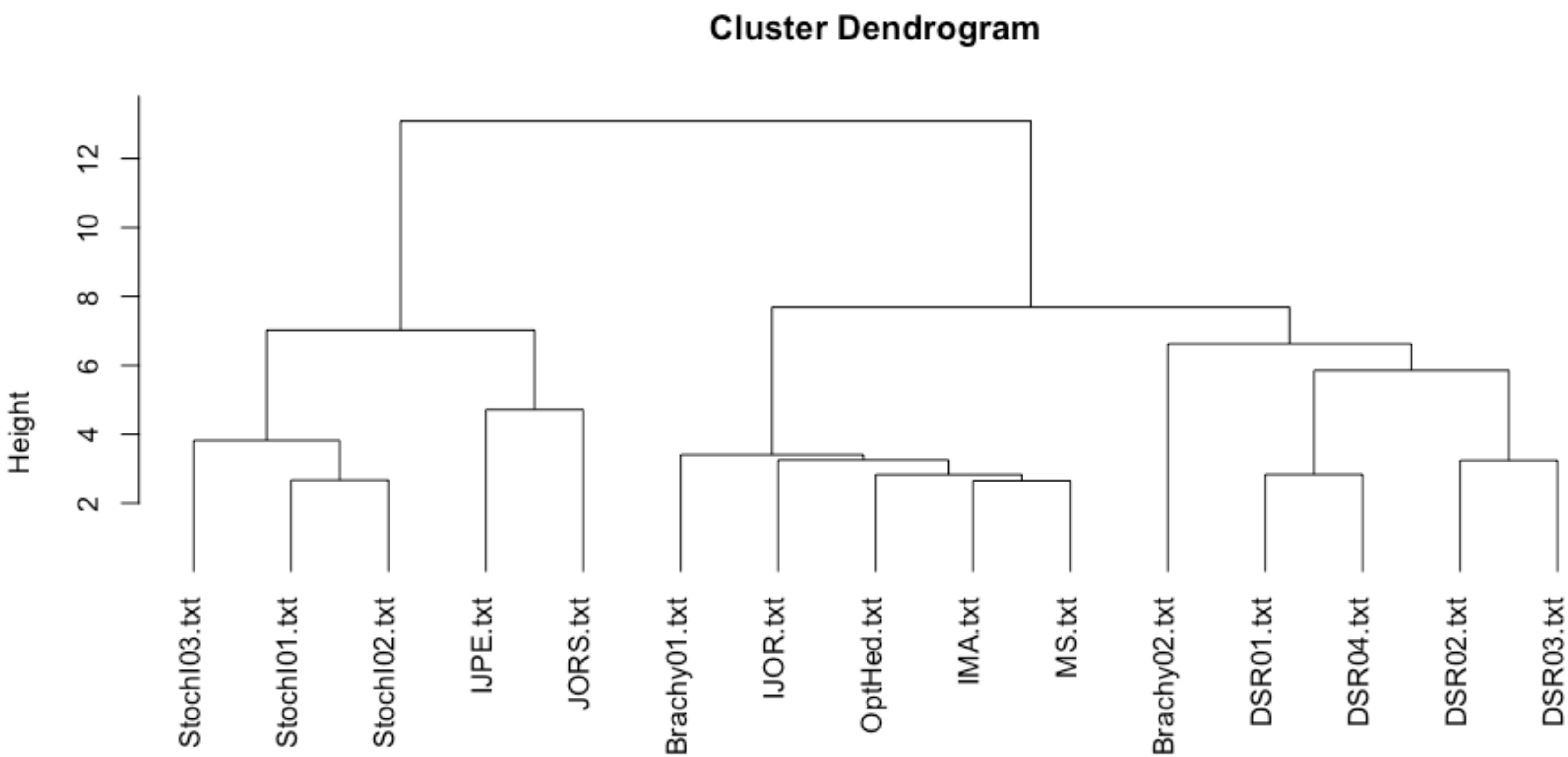
R: Term-Document Matrix

	base	can	cost	effici	high	inventori	level	method	model	paper	reduc	thus	use
Brachy01.	0	1	0	0	1	0	0	0	0	0	1	0	1
Brachy02.	3	0	0	0	2	0	0	0	2	0	2	0	3
DSR01.txt	0	0	4	0	1	0	0	1	0	1	3	0	2
DSR02.txt	0	0	1	1	0	0	0	3	0	0	1	0	3
DSR03.txt	1	0	3	0	0	0	0	4	0	0	1	1	1
DSR04.txt	0	0	4	2	1	0	0	1	0	1	2	0	1
IJOR.txt	1	1	2	0	0	2	1	0	1	1	0	0	2
IJPE.txt	0	2	2	1	0	5	1	1	0	2	1	1	1
IMA.txt	0	0	0	0	0	2	0	0	2	1	1	1	1
JORS.txt	0	1	2	1	0	8	2	1	1	0	1	2	4
MS.txt	0	0	0	0	0	1	0	0	2	0	0	0	0
OptHed.tx	1	0	2	0	0	0	1	0	4	1	1	0	0
StochI01.t	1	1	0	2	1	6	6	2	7	0	0	1	1
StochI02.t	2	1	0	2	1	5	5	2	6	1	0	1	3
StochI03.t	1	2	3	1	1	3	4	1	5	0	0	1	2

R: Save TDM, Cluster

```
> # ref Williams
> tdms = as.matrix(dtms)
> write.csv(tdms, ”tdms.csv")
> distmatrix = dist(scale(tdms))
> # note that we are not using cosine distance!
> fit = hclust(distmatrix, method = "ward.D")
> plot(fit)
> plot(fit, hang = -1)
```

R: Plot dendrogram



Summary

Text analytics

- Overview
- Processing text for analysis
- Representing text by a Term-Document Matrix
- Weighting factors for document distance calculations

Text analytics in R

- Text processing
- Document clustering

Review Question Answers

1. B
2. F
3. A
4. E

Notes on the presentation

This presentation contains slides created to accompany: *Introduction to Data Mining*, Tan, Steinbach, Kumar. Pearson Education Inc., 2006.

Additional material from: *Practical Text Mining and Statistical Analysis for Non-structured Text Data Applications*, Miner, G. et al., Elsevier, 2012.

Presentation originally created by Dr. Sue Bedingfield, with additions by Rui Jie Chow & Dr. Parthan Kasarapu.