ACDS Lecture Series

Lecture - 13

Text Mining

G. N. Sastry and Team

ADVANCED COMPUTATION AND DATA SCIENCES (ACDS) DIVISION

CSIR-North East Institute of Science and Technology, Jorhat, Assam, India

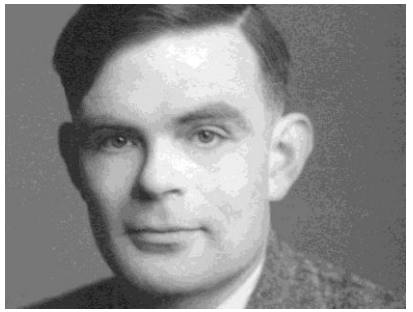
13. Text Mining

The Topics Covered in This Section

13.1.1 What is Text Mining and NLP
13.1.2 Text Mining vs. NLP
13.1.3 Applications
13.1.4 Challenges
13.1.5 Requirements and Setup

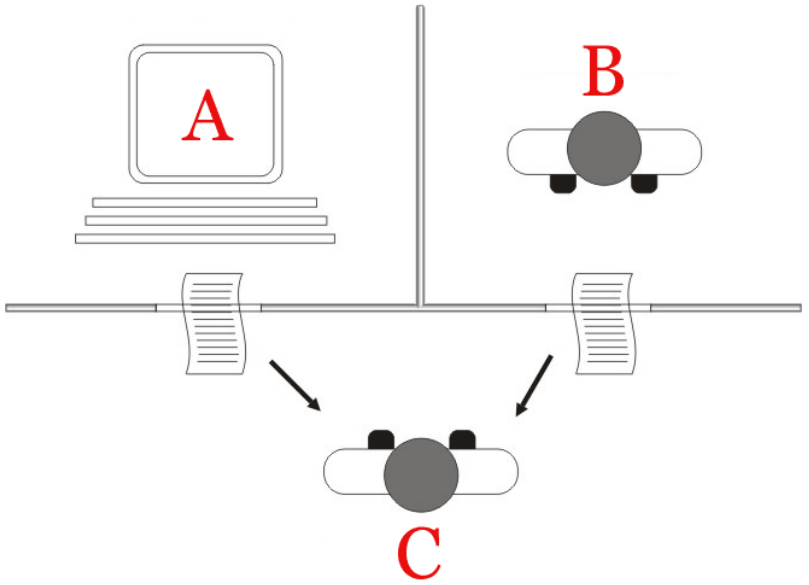
13.1.1 What is Text Mining and NLP?

Can a human communicate to an AI system in natural language?



> The original question, "Can machines think?", I believe to be too meaningless to deserve discussion. Nevertheless I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted.
>
> -Alan Turing (1950)

"Computing Machinery and Intelligence"



Challenges:

- Computers need structured data, but human language is unstructured and ambiguous in nature.

13.1.1 What is Text Mining and NLP?

- NLP and text mining are usually used for different goals.
- There is a difference in methods also.
- Although, there is indeed an overlap and both definitions are vogue.

Text Mining

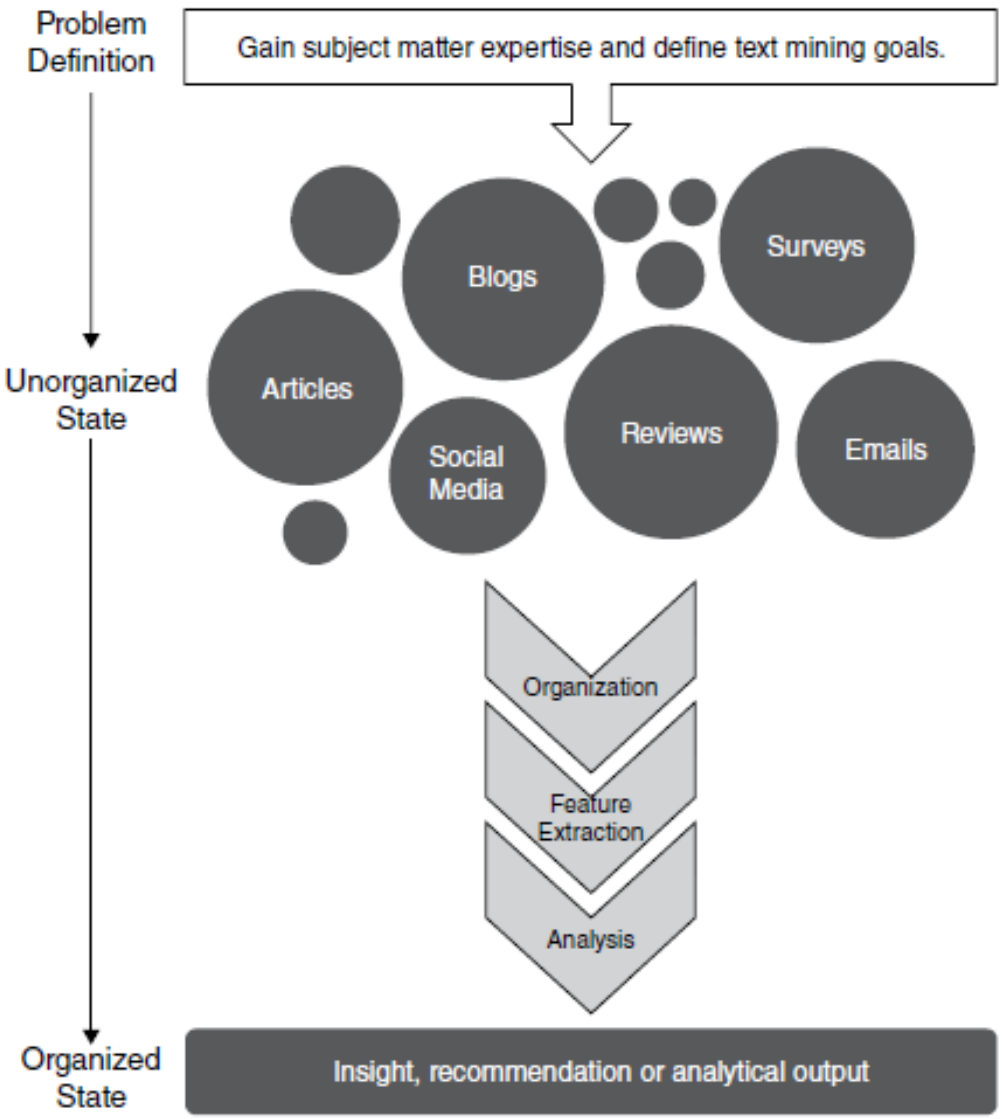
- Text mining is "the discovery of new, previously unknown information, by automatically extracting information from different written resources."
- Tasks include patterns matching, text categorization, text clustering, concept/entity extraction, sentiment analysis, document summarization, and entity relation modeling.

Natural Language Processing (NLP)

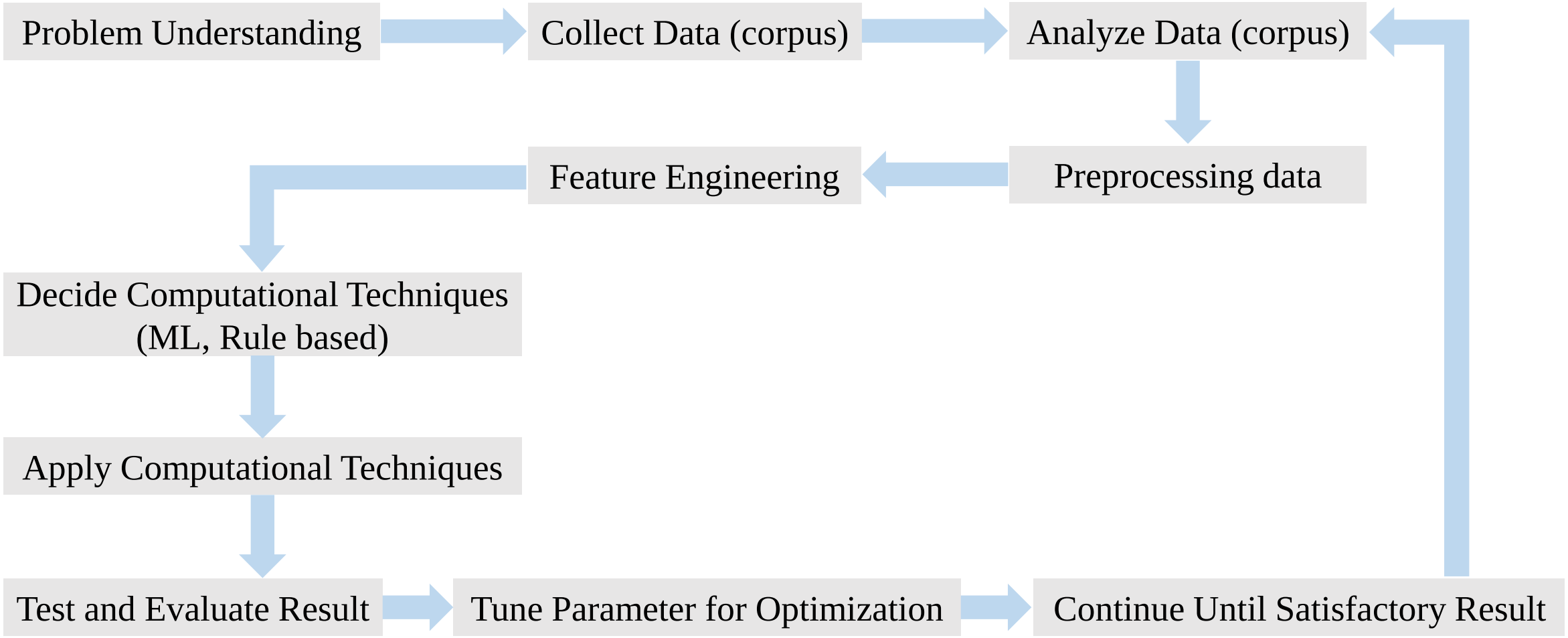
- NLP is a subfield of computer science, information engineering, and artificial intelligence.
- Task include speech recognition, natural language understanding, and natural language generation.

Basic Workflow – How the Process Works ?

- Define the problem and specific goals
- Identify the text that needs to be collected
- Organize the text
- Extract features
- Analyze
- Reach an insight or recommendation



Workflow in Detail



13.1.2 Text Mining vs. NLP

Goals and Tasks	
Text Mining	**Natural Language Processing**
• Deals with text itself (no semantic meaning). • Speech recognition systems is not a part.	• Deals with underlying metadata. • Speech recognition systems is a part of NLP.
• Extract useful information from the structured and unstructured text data. Like, • Patterns in text. • Matching structure. • Frequency counts of words. • Presence/absence of certain words.	• Recognize and understand language by processing and analyzing Languages. Like, • Looking at the semantics in the text. • Analyzing grammatical structures. • Sentiments. • Document classification and clustering.

13.1.2 Text Mining vs NLP

Techniques and Tools	
Text Mining	**Natural Language Processing**
• Techniques are usually shallow and do not consider the text structure. Like, • Use bag of words. • n-grams and stemming. • Correlation of words.	• Techniques usually involve the test structure (grammatical/lexical relations). Like, • Sentence splitting • Part of speech tagging • Parse tree construction.
• Statistical ML models. Like, • Perl, R tools.	• Advanced ML Models - Deep Learning. Like, • Toolkits Like NLTK in Python.

13.1.2 Text Mining vs NLP

Example:

Consider a message –

> I am not happy. I will be happy when it will rain. If it will rain, I'll be happy. She asked whether I am happy. Are you happy?

- Text mining method will consider the message to indicate happiness.
- While typical NLP methods detects that its not just about happiness.

- Text Mining is just like Data Mining.
- NLP is the bigger fish which uses text-mining.
- However people use the terms interchangeably often but there is a distinction boundary clearly.

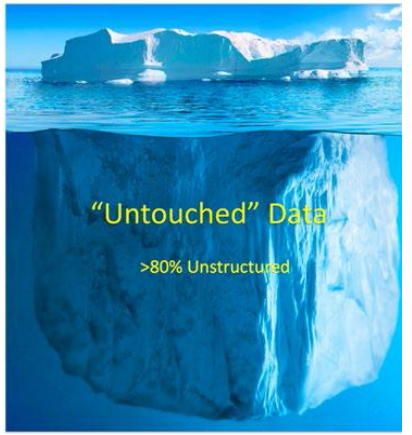
13.1.3 Applications

- Keyword Searching
- Information Extraction
- Document Summarization
- Document Classification
- Document Clustering
- Topic Modeling and Segmentation
- Sentiment Analysis
- Question Answering System
- Template-based Chatbots
- Language Translation
- Speech Recognition System
- And so on……..

Ambiguity: Hospitals Are Sued by 7 Foot Doctors.
World Knowledge: … a mutation on the *for* gene …
Informal: Its too goooooooood…….♥

13.1.4 Challenges

- Data challenges
- Unstructured form
- Long term dependencies
- Word Ambiguity
- Context dependencies
- Sarcasm involvement
- Misspellings in entity extraction
- And so on……..



13.1.5 Requirements and Set Up

Prerequisites

- R or Python Programming
- Probability and Statistics
- Linear Algebra
- Machine Learning and Deep Learning Concepts

Set Up

- Anaconda and Python Packages
- Rstudio and R Packages

install.packages (c("tm")
library(tm)

tm
pdftools
Wordcloud
Rstem
Tokenizers
SnowballC

spacy and nltk
panda, numpy, scipy, matplotliub, seaborn
sklearn, keras, tensorflow

conda install scikit-learn
pip install scikit-learn
import sklearn as sk

13.1.5 Requirements and Set Up

	SPACY	SYNTAXNET	NLTK	CORENLP
Programming language	Python	C++	Python	Java
Neural network models	✓	✓	✗	✓
Integrated word vectors	✓	✗	✗	✗
Multi-language support	✓	✓	✓	✓
Tokenization	✓	✓	✓	✓
Part-of-speech tagging	✓	✓	✓	✓
Sentence segmentation	✓	✓	✓	✓
Dependency parsing	✓	✓	✗	✓
Entity recognition	✓	✗	✓	✓
Coreference resolution	✗	✗	✗	✓

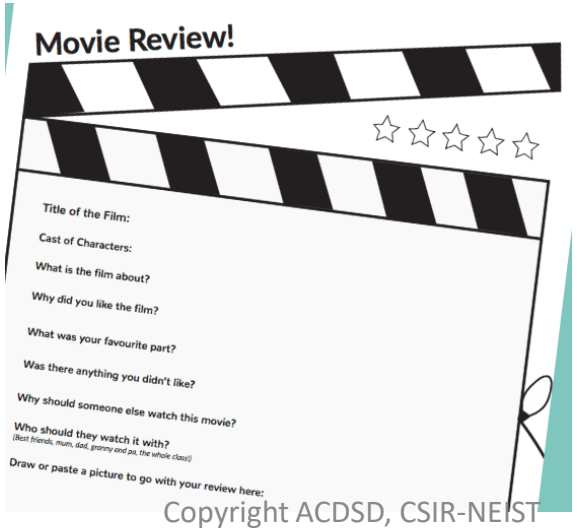
	ABSOLUTE (MS PER DOC)			RELATIVE (TO SPACY)		
SYSTEM	TOKENIZE	TAG	PARSE	TOKENIZE	TAG	PARSE
spaCy	0.2ms	1ms	19ms	1x	1x	1x
CoreNLP	0.18ms	10ms	49ms	0.9x	10x	2.6x
ZPar	1ms	8ms	850ms	5x	8x	44.7x
NLTK	4ms	443ms	*n/a*	20x	443x	*n/a*

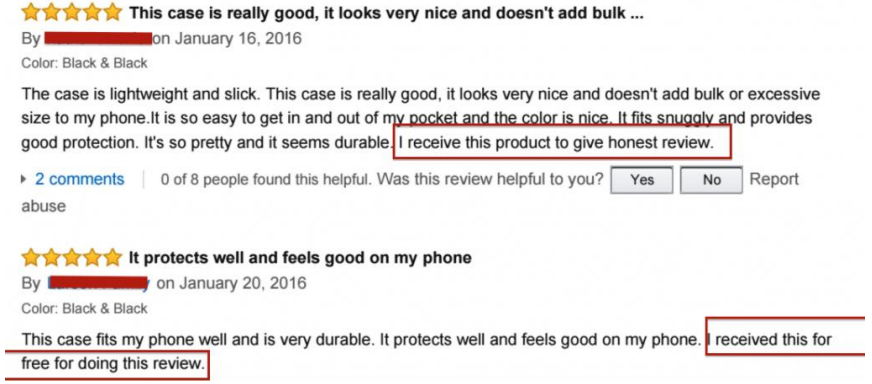
The Topics Covered in This Section

13.2.1 Tokenization
13.2.2 Ngram Model
13.2.3 Stemming and Lemmatization
13.2.4 Cleaning
13.2.5 Bag of Words
13.2.6 TF and IDF
13.2.7 POS tagging
13.2.8 Named Entity Recognition
13.2.9 Syntax Parsing
13.2.10 Chunking and Chinking

The sources of data (text)

- Pdf Files (.pdf)
- Text Files (.txt)
- Websites and Blogs (https://)
- Social Media (fb, Twitter)
- Articles (Research Article, Survey)
- Review (Amazon, Movie Review)
- Emails (@gmail)





13.2.1 Tokenization& Segmentation

- Humans we can tell there is a plethora of information inside of text documents.
- But a computer needs specialized processing techniques in order to “understand” raw text data.
- Text data is highly unstructured and can be in multiple languages!
- Natural Language Processing attempts to use a variety of techniques in order to create structure out of text data.

Text Wrangling Includes

- Tokenization & Segmentation
- Stemming and Lemmatization
- Cleaning like removal numbers, punctuations, case etc.

Tokenization & Segmentation

- Tokenization is the process of breaking up the original text into component pieces (tokens).
- Notice that tokens are pieces of the original text that helps in understand the meaning of the text and their relationship to one another.

For Example, break a complex sentence into token of words.

Text	→	**This is an ACDS Lecture.**
Tokens	→	**“This”, “is”, “an”, “ACDS”, “Lecture”, “.”**

Zipf's law

- A Model used to understand the distribution of terms - how terms are distributed across documents.
- It states that, if t_1 is the most common term in the collection, t_2 is the next most common, and so on, then the collection frequency cf_i of the i_{th} most common term is proportional to 1/i
- So if the most frequent term occurs cf_1 times, then the second most frequent term has half as many occurrences, the third most frequent term a third as many occurrences, and so on.
- It is therefore a *power law* with exponent.

Issues in Tokenization

- Finland's capital → Finland Finlands Finland's ?
- what're, I'm, isn't → What are, I am, is not
- Hewlett-Packard → Hewlett Packard ?
- state-of-the-art → state of the art ?
- Lowercase → lower-case lowercase lower case ?
- San Francisco → one token or two?
- m.p.h., PhD. → ??

Probabilistic Language Models

- **Goal:** compute the probability of a sentence or sequence of words:

  $P(W) = P(w_1,w_2,w_3,w_4,w_5 \ldots w_n)$

- **Related task:** probability of an upcoming word:

  $P(w_5|w_1,w_2,w_3,w_4)$

- A model that computes either of these:

  $P(W)$ or $P(w_n|w_1,w_2 \ldots w_{n-1})$ is called a **language model**.

Importance:

- Machine Translation: P(**high** winds tonite) > P(**large** winds tonite)
- Spell Correction: P(about fifteen **minutes** from) > P(about fifteen **minuets** from)
- Speech Recognition: P(I saw a van) >> P(eyes awe of an)
- Summarization, question-answering, etc., etc.!!

How to compute P(W)

- The Chain Rule of Probability is applied to compute joint probability of words in sentence

$$P(x_1,x_2,x_3,\ldots,x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1,x_2)\ldots P(x_n|x_1,\ldots,x_{n-1})$$

$$P(W_1\ W_2..W_n) = \prod_i P(W_i|W_1W_2..W_{i-1})$$

Example:

P("its water is so transparent") =

$$P(\text{its}) \times P(\text{water}|\text{its}) \times P(\text{is}|\text{its water})$$
$$\times P(\text{so}|\text{its water is}) \times P(\text{transparent}|\text{its water is so})$$

- Could we just count and divide?

$$P(\text{the} \mid \text{its water is so transparent that}) = \frac{Count(\text{its water is so transparent that the})}{Count(\text{its water is so transparent that})}$$

- No! Too many possible sentences!
- We'll never see enough data for estimating these

Markov Assumption (Simplifying assumption):

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{that})$$

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{transparent that})$$

- In other words, we approximate each component in the product

$$P(W_1\, W_2 .. W_n) \approx \prod_i P(W_i | W_{i-k} .. W_{i-1})$$

$$P(W_i | W_1\, W_2 .. W_n) \approx \prod_i P(W_i | W_{i-k} .. W_{i-1})$$

13.2.2 Ngram Model

Ngrams Model: Tokens of any number of consecutive written words

- Unigram: (simplest Markov model)) $P(W_1|W_2..W_n) \approx \prod P(W_i)$
- Bigram: (condition on the previous word) $P(W_i|W_1\ W_2..W_{i-1}) \approx P(W_i|W_{i-1})$
- We can extend to trigrams, 4-grams, 5-grams

Example:

This is an ACDS Lecture

Uni-gram	This	is	an	ACDS	Lecture

Bi-gram	This is	is an	an ACDS	ACDS Lecture

Tri-gram	This is an	is an ACDS	an ACDS Lecture

Estimating N-gram

- The Maximum Likelihood Estimate

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

<s> I am Sam </s>

<s> Sam I am </s>

<s> I do not like green eggs and ham </s>

$$P(\texttt{I} \mid \texttt{<s>}) = \tfrac{2}{3} = .67 \qquad P(\texttt{Sam} \mid \texttt{<s>}) = \tfrac{1}{3} = .33 \qquad P(\texttt{am} \mid \texttt{I}) = \tfrac{2}{3} = .67$$

$$P(\texttt{</s>} \mid \texttt{Sam}) = \tfrac{1}{2} = 0.5 \qquad P(\texttt{Sam} \mid \texttt{am}) = \tfrac{1}{2} = .5 \qquad P(\texttt{do} \mid \texttt{I}) = \tfrac{1}{3} = .33$$

Analysis of N gram Model:

- In general this is an insufficient model of language
- Because language has long-distance dependencies.

  "The computer which I had just put into the machine room on the fifth floor crashed."
- But we can often get away with N-gram models

13.2.3 Stemming and Lemmatization

Normalization

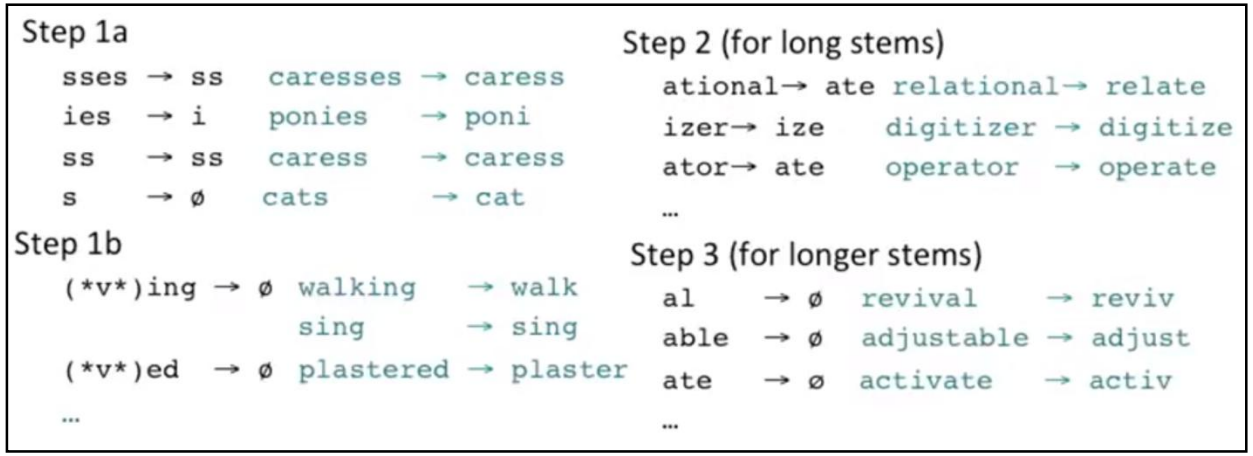
- **Stemming:** Normalize word to base form by chopping off letters from the end
- **Lemmatization:** Meaningful stemming considering language's full vocabulary

Word	Stemming	Lemmatization
was	wa	be
studies	studi	Study
studying	Study	study

- **Porter Stemmer**

➢ Developed by Martin Porter in 1980

➢ The popular English Stemmer



- **Snowball Stemmer**

➢ Offers a slight improvement over the original Porter stemmer, both in logic and speed.

Note: Lemmatization is typically seen as more informative than stemming.

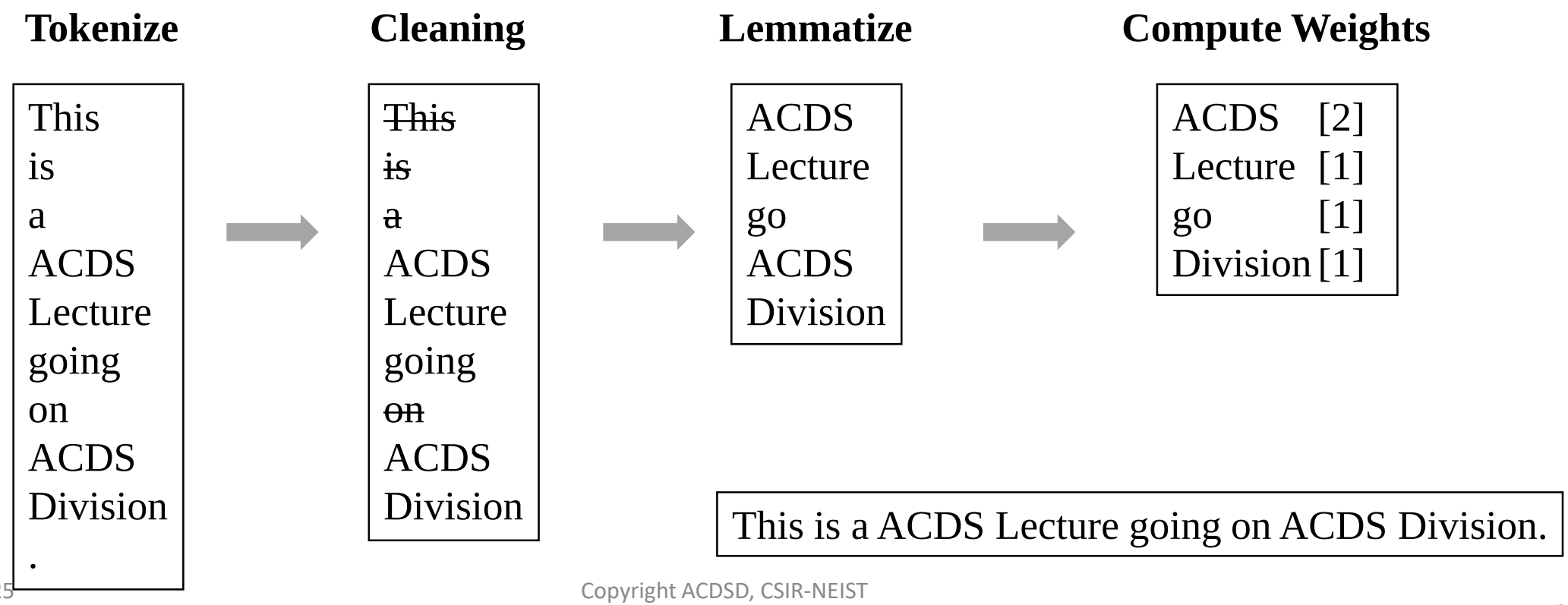
Cleaning

- Case Folding
- Special Character Removal
- Punctuation Removal
- Stopwords Removal
- Number Removal
- Whitespace Removal
- Frequent Words Removal
- Rare Words Removal
- Spelling Correction

Words like "a" and "the" appear so frequently that don't require tagging as thoroughly as nouns, verbs and modifiers.

13.2.5 Bag of Words

- Most classic machine learning algorithms can't take in raw text.
- Need to perform a feature "extraction" from the raw text in order to pass numerical features to the machine learning algorithm.
- For example, count the occurence of each word to map text to a number.



Document term matrix

- In order to find the similarity between documents in a corpus, we can use a document term matrix.
- In a document term matrix, rows represent documents, columns represent terms, and each cell value is the term frequency count for a document.

Document-1: Ice creams in summer are awesome **Document-2**: I love ice creams in summer **Document-3**: Ice creams are awesome all season

	Ice cream	Summer	Love	Awesome	season
Doc1	1	1	0	1	0
Doc2	1	1	1	0	0
Doc3	1	0	0	1	1

- If we visualize this in a term-document space, each document becomes a point in it.
- Similar among documents calculated by the (Euclidean) distance between the two points.
- These frequently occurring terms can affect the similarity comparison.
- The term space will be biased towards these terms.
- In order to address this problem, we use inverse document frequency.

13.2.6 TF and IDF

- **term frequency** tf(*t*,*d*): the number of times that term *t* occurs in document *d* upon total terms w in the document.
- **inverse document frequency** idf(t, D): the number of times that term *t* occurs across all documents D upon total documents |D|.
- **tf-idf**: product of tf(t,d) and idf(t,D)

- tf alone isn't enough! as terms, like "a" or "the" which are so common, tf will tend to incorrectly emphasize documents, without giving enough weight to more meaningful terms.
- idf factor is incorporated which diminishes the weight of terms that occur very frequently in the document set and increases the weight of terms that occur rarely.
- TF-IDF allows to understand the context of words across an entire corpus of documents, instead of just its relative importance in a single document unlike DTM or TF.

$$\mathbf{tf}(t, d) = \frac{f_d(t)}{\max_{w \in d} f_d(w)}$$

$$\mathbf{idf}(t, D) = \ln\left(\frac{|D|}{|\{d \in D : t \in d\}|}\right)$$

$$\mathbf{tfidf}(t, d, D) = \mathbf{tf}(t, d) \cdot \mathbf{idf}(t, D)$$

$$\mathbf{tfidf}'(t, d, D) = \frac{\mathbf{idf}(t, D)}{|D|} + \mathbf{tfidf}(t, d, D)$$

$f_d(t) :=$ frequency of term t in document d

$D :=$ corpus of documents

Example:

$$\text{tf}(\text{"this"}, d_1) = \frac{1}{5} = 0.2$$
$$\text{tf}(\text{"this"}, d_2) = \frac{1}{7} \approx 0.14 \qquad \text{idf}(\text{"this"}, D) = \log\left(\frac{2}{2}\right) = 0$$

$$\text{tfidf}(\text{"this"}, d_1, D) = 0.2 \times 0 = 0$$
$$\text{tfidf}(\text{"this"}, d_2, D) = 0.14 \times 0 = 0$$

Document 1

Term	Term Count
this	1
is	1
a	2
sample	1

Document 2

Term	Term Count
this	1
is	1
another	2
example	3

$$\text{tf}(\text{"example"}, d_1) = \frac{0}{5} = 0$$
$$\text{tf}(\text{"example"}, d_2) = \frac{3}{7} \approx 0.429$$
$$\text{idf}(\text{"example"}, D) = \log\left(\frac{2}{1}\right) = 0.301$$

$$\text{tfidf}(\text{"example"}, d_1, D) = \text{tf}(\text{"example"}, d_1) \times \text{idf}(\text{"example"}, D) = 0 \times 0.301 = 0$$
$$\text{tfidf}(\text{"example"}, d_2, D) = \text{tf}(\text{"example"}, d_2) \times \text{idf}(\text{"example"}, D) = 0.429 \times 0.301 \approx 0.129$$

13.2.6 TF and IDF

Variants of term frequency (tf) weight

weighting scheme	tf weight
binary	$0, 1$
raw count	$f_{t,d}$
term frequency	$f_{t,d} \Big/ \sum_{t' \in d} f_{t',d}$
log normalization	$\log(1 + f_{t,d})$
double normalization 0.5	$0.5 + 0.5 \cdot \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$
double normalization K	$K + (1 - K) \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$

Variants of inverse document frequency (idf) weight

weighting scheme	idf weight ($n_t = \lvert\{d \in D : t \in d\}\rvert$)
unary	1
inverse document frequency	$\log \frac{N}{n_t} = -\log \frac{n_t}{N}$
inverse document frequency smooth	$\log\left(\frac{N}{1+n_t}\right) + 1$
inverse document frequency max	$\log\left(\frac{\max_{\{t' \in d\}} n_{t'}}{1+n_t}\right)$
probabilistic inverse document frequency	$\log \frac{N - n_t}{n_t}$

Recommended tf–idf weighting schemes

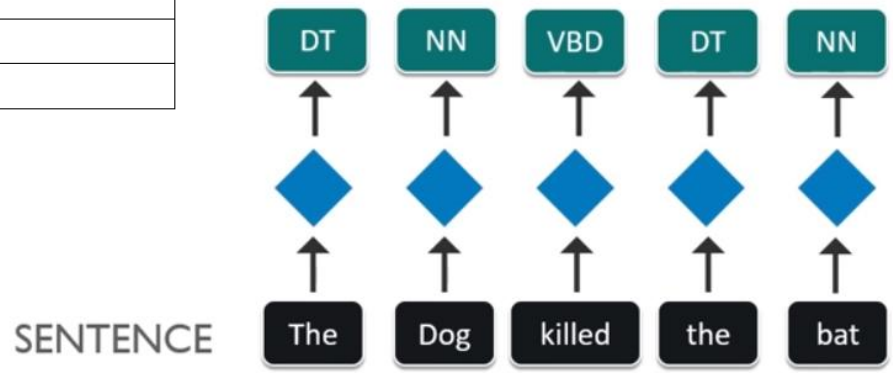
weighting scheme	document term weight	query term weight
1	$f_{t,d} \cdot \log \frac{N}{n_t}$	$\left(0.5 + 0.5 \frac{f_{t,q}}{\max_t f_{t,q}}\right) \cdot \log \frac{N}{n_t}$
2	$1 + \log f_{t,d}$	$\log\left(1 + \frac{N}{n_t}\right)$
3	$(1 + \log f_{t,d}) \cdot \log \frac{N}{n_t}$	$(1 + \log f_{t,q}) \cdot \log \frac{N}{n_t}$

- Used by search engines as a central tool in scoring and ranking a document's relevance given a user query.
- Used for stop-words filtering in various subject fields, including text summarization and classification.

13.2.7 PoS Tagging

Tag	Description
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending
PRP	Personal pronoun

Tag	Description
PRP$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	to
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Whdeterminer
WP	Whpronoun
WP$	Possessive whpronoun
WRB	Whadverb



13.2.8 Named Entity Recognition

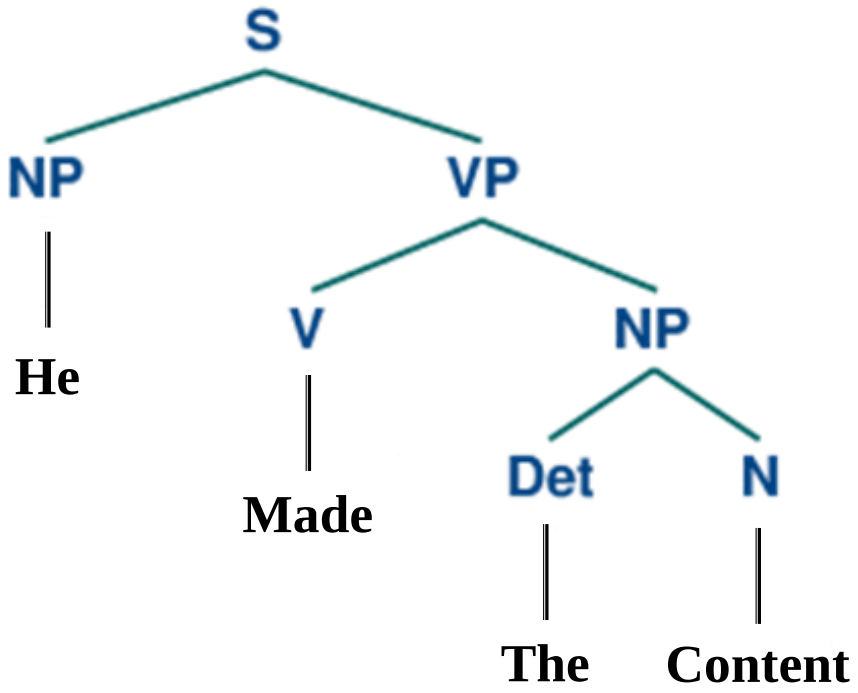
- NER seeks to locate and classify named entity mentions in unstructured text into pre-defined categories such as the person names, organizations, locations, medical codes, time expressions, quantities, monetary values, percentages, etc.

For Example:

- Jim bought 300 shares of Acme Corp. in 2006.
- $[\text{Jim}]_{\textbf{Person}}$ bought 300 shares of $[\text{Acme Corp.}]_{\textbf{Organization}}$ in $[2006]_{\textbf{Time}}$.

- The decision by the independent MP Andrew Wilkie to withdraw his support for the minority Labor government sounded dramatic but it should not further threaten its stability. When, after the 2010 election, Wilkie, Rob Oakeshott, Tony Windsor and the Greens agreed to support Labor, they gave just two guarantees: confidence and supply.

Person
Date
Location
Organization



13.2.10 Chunking and Chinking

- Using modified regular expressions, we can define **chunk patterns**.
- These are patterns of part-of-speech tags that define what kinds of words make up a chunk.

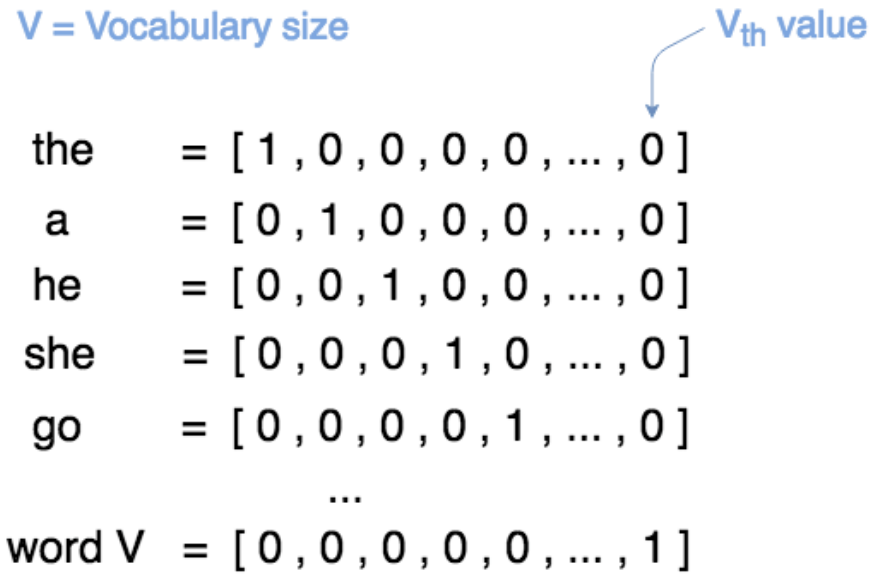
- We can also define patterns for what kinds of words should not be in a chunk.
- These unchunked words are known as **chinks**.

The Topics Covered in This Section

13.3.1 Matrix Factorization
13.3.2 Word2vec, Text2vec, Glove, Paragraph2vec
13.3.3 Word Similarity
13.3.4 Continuous Bag of Words
13.3.6 Exercises

13.3.1 Matrix Factorization

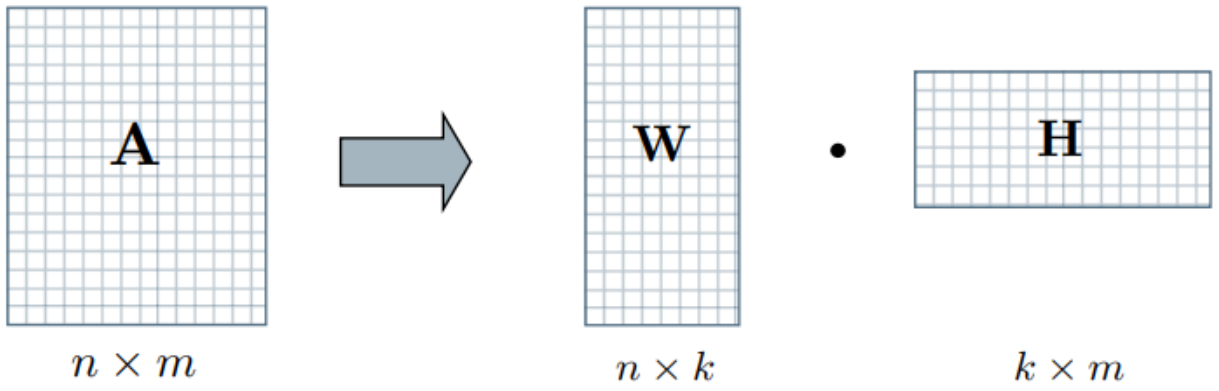
- A popular idea in modern machine learning is to represent words by vectors.
- These vectors capture hidden information about a language, like word analogies or semantic.
- It is also used to improve performance of text classifiers.
- **Text Representation:** Represent text in vector form for computer understanding.



13.3.1 Matrix Factorization

Non-negative Matrix Factorization

- An unsupervised algorithm that simultaneously performs dimensionality reduction and clustering.
- We can use it in conjunction with TF-IDF to model topics across documents.
- Given a non-negative matrix A, find non-negative factors W and H of k-dimension.
- Approximate each object (i.e. column of A) by a linear combination of k reduced dimensions.
- Each basis vector can be interpreted as a cluster.
- The memberships of objects in these clusters encoded by H.



$$\mathbf{W} \geq 0 \ , \ \mathbf{H} \geq 0$$

- Basis vectors: the topics (clusters) in the data.
- Coefficient matrix: the membership weights for documents relative to each topic (cluster).

13.3.1 Matrix Factorization

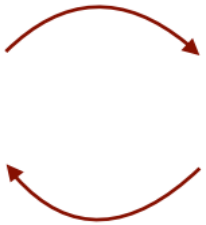
- **Objective Function:** Some measure of reconstruction error between A and the approximation WH

$$\frac{1}{2}||\mathbf{A} - \mathbf{WH}||_{\mathrm{F}}^2 = \sum_{i=1}^{n}\sum_{j=1}^{m}(A_{ij} - (WH)_{ij})^2$$

- Refine W and H in order to minimize the objective function.
- Common approach is to iterate between two multiplicative update rules until convergence

1. Update **H**

$$H_{cj} \leftarrow H_{cj}\frac{(W\mathbf{A})_{cj}}{(W\mathbf{WH})_{cj}}$$

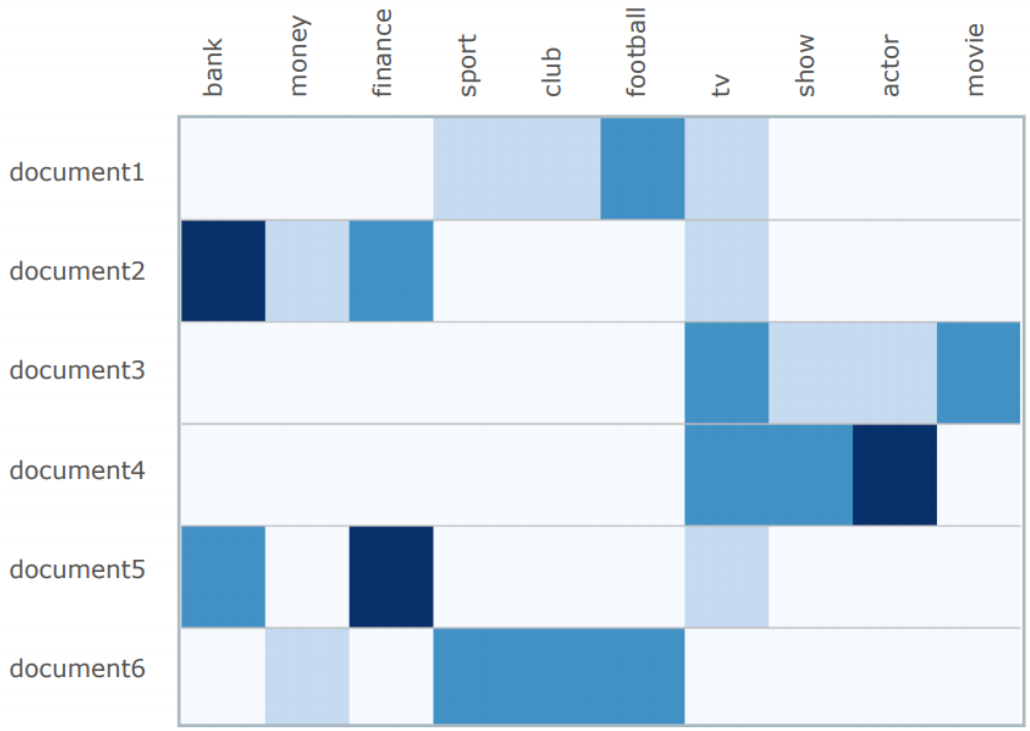


2. Update **W**

$$W_{ic} \leftarrow W_{ic}\frac{(\mathbf{A}H)_{ic}}{(\mathbf{WH}H)_{ic}}$$

13.3.1 Matrix Factorization

- Construct vector space model for documents (after stop word filtering), resulting in a term-document matrix A.
- Apply TF-IDF term weight normalization to A
- Normalize TF-IDF vectors to unit length.
- Initialize factors using NNDSVD on A.
- Apply Projected Gradient NMF to A.



- Just like LDA, we will need to select the number of expected topics beforehand (the value of **k**)!
- Just like LDA, we have to interpret the topics based on coefficient values of the words per topic.

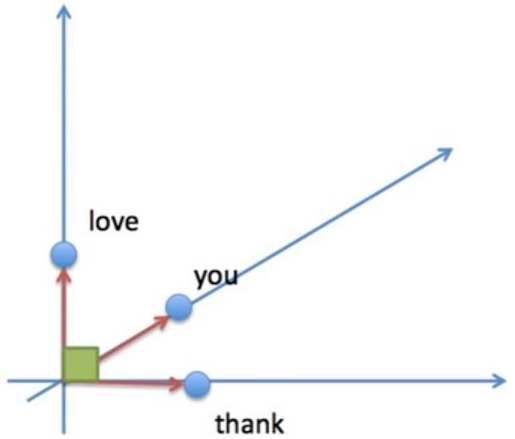
Word Embedding: Dense vector for vector representation of word with similarity.

What is Word2vec?

- A shallow two-layer neural network model trained to produce word embeddings.
- Takes a large corpus of text as its input and produces a vector space, typically of several hundred dimensions, with each unique word being assigned a corresponding vector in the space.
- Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located close to one another in the space.

Why do we need word2vec?

- One hot encoding does not have similarity.
- Every distance is same to each other.
- Cosine similarity is also 0 as angle is 90 degree.

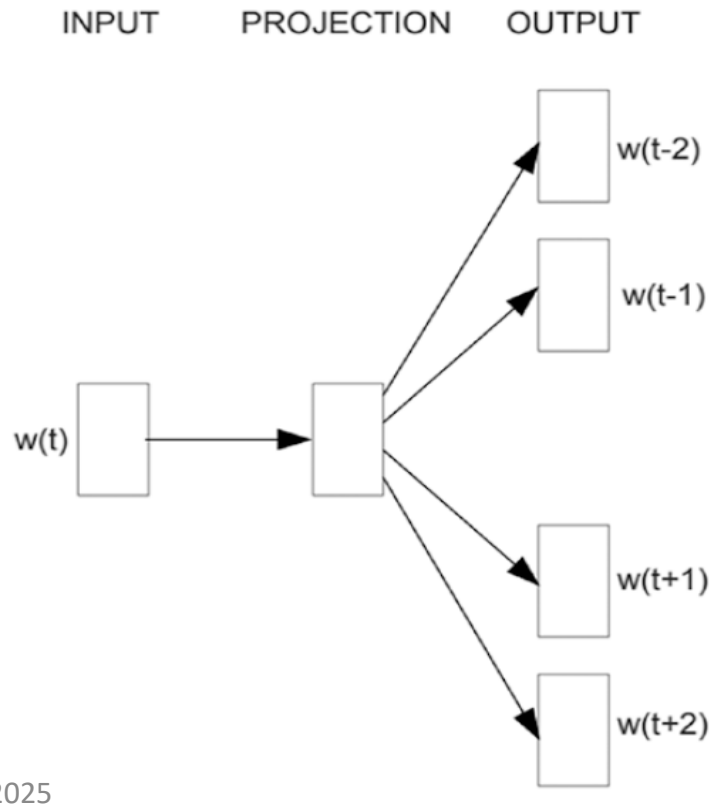


unique word	encoding
thank	[1, 0, 0]
you	[0, 1, 0]
love	[0, 0, 1]

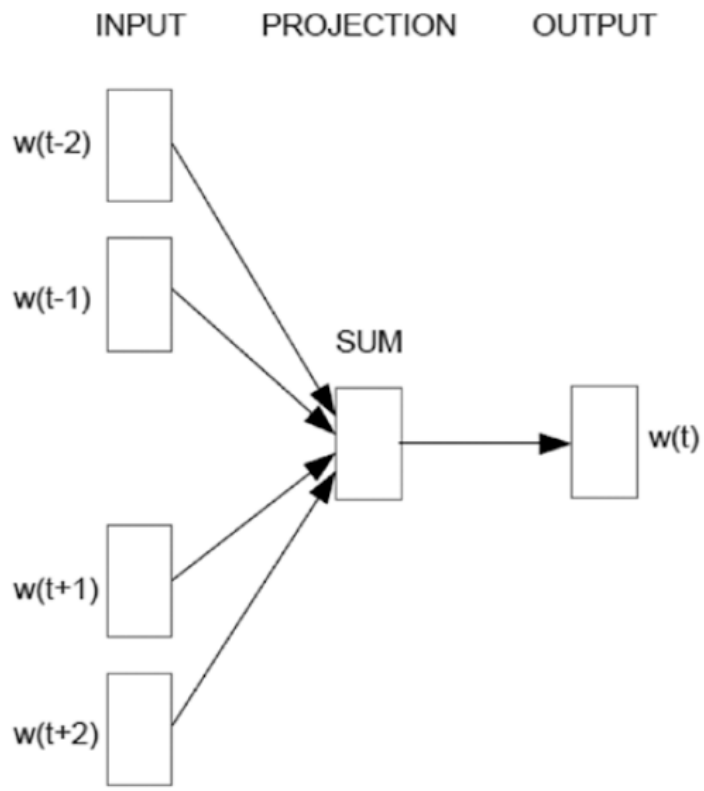
13.3.2 Word2vec, Text2vec, GloVe, Paragraph2vec

- Word2vec trains words against other words that neighbor them in the input corpus.
- Word2vec can be achieved by two approaches -

Skipgram
using a word to predict a target context



Continuous bag of words (CBOW)
using context to predict a target word



- **Word2vec data generation using skipgram**

"king brave man"
"queen beautiful woman"

Window size 1

word	neighbor
king	brave
brave	king
brave	man
man	brave
queen	beautiful
beautiful	queen
beautiful	woman
woman	beautiful

Window size 2

word	neighbor
king	brave
king	man
brave	king
brave	man
man	king
man	brave
queen	beautiful
queen	woman
beautiful	queen
beautiful	woman
woman	queen
woman	beautiful

word	word one hot encoding	neighbor	neighbor one hot encoding
king	[1, 0, 0, 0, 0, 0]	brave	[0, 1, 0, 0, 0, 0]
king	[1, 0, 0, 0, 0, 0]	man	[0, 0, 1, 0, 0, 0]
brave	[0, 1, 0, 0, 0, 0]	king	[1, 0, 0, 0, 0, 0]
brave	[0, 1, 0, 0, 0, 0]	man	[0, 0, 1, 0, 0, 0]
man	[0, 0, 1, 0, 0, 0]	king	[1, 0, 0, 0, 0, 0]
man	[0, 0, 1, 0, 0, 0]	brave	[0, 1, 0, 0, 0, 0]
queen	[0, 0, 0, 1, 0, 0]	beautiful	[0, 0, 0, 0, 1, 0]
queen	[0, 0, 0, 1, 0, 0]	woman	[0, 0, 0, 0, 0, 1]
beautiful	[0, 0, 0, 0, 1, 0]	queen	[0, 0, 0, 1, 0, 0]
beautiful	[0, 0, 0, 0, 1, 0]	woman	[0, 0, 0, 0, 0, 1]
woman	[0, 0, 0, 0, 0, 1]	queen	[0, 0, 0, 1, 0, 0]
woman	[0, 0, 0, 0, 0, 1]	beautiful	[0, 0, 0, 0, 1, 0]

- **Thus, Word2vec deep learning model input and target**

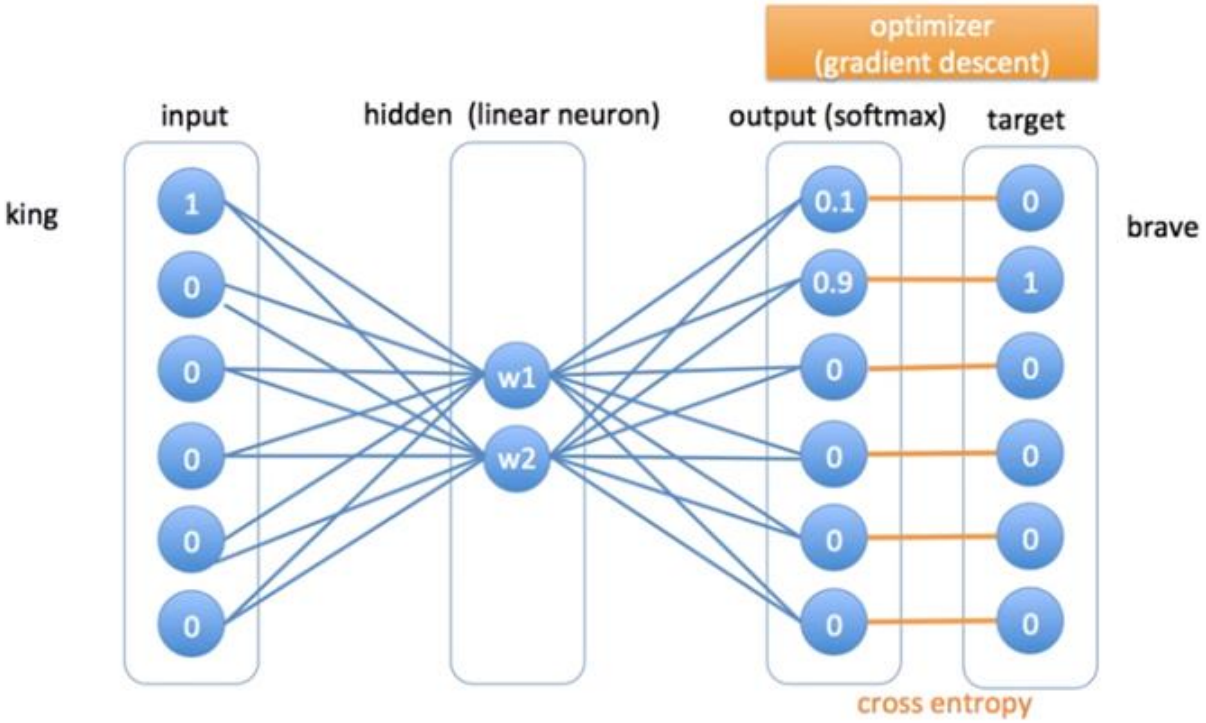
"king brave man"
"queen beautiful woman"

word	word one hot encoding	neighbor	neighbor one hot encoding
king	[1, 0, 0, 0, 0, 0]	brave	[0, 1, 0, 0, 0, 0]
king	[1, 0, 0, 0, 0, 0]	man	[0, 0, 1, 0, 0, 0]
brave	[0, 1, 0, 0, 0, 0]	king	[1, 0, 0, 0, 0, 0]
brave	[0, 1, 0, 0, 0, 0]	man	[0, 0, 1, 0, 0, 0]
man	[0, 0, 1, 0, 0, 0]	king	[1, 0, 0, 0, 0, 0]
man	[0, 0, 1, 0, 0, 0]	brave	[0, 1, 0, 0, 0, 0]
queen	[0, 0, 0, 1, 0, 0]	beautiful	[0, 0, 0, 0, 1, 0]
queen	[0, 0, 0, 1, 0, 0]	woman	[0, 0, 0, 0, 0, 1]
beautiful	[0, 0, 0, 0, 1, 0]	queen	[0, 0, 0, 1, 0, 0]
beautiful	[0, 0, 0, 0, 1, 0]	woman	[0, 0, 0, 0, 0, 1]
woman	[0, 0, 0, 0, 0, 1]	queen	[0, 0, 0, 1, 0, 0]
woman	[0, 0, 0, 0, 0, 1]	beautiful	[0, 0, 0, 0, 1, 0]

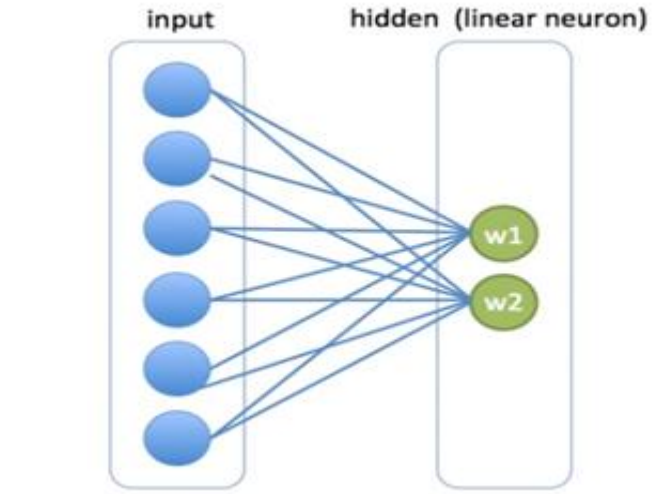
input (word one hot encoding)	target (neighbor one hot encoding)
[1, 0, 0, 0, 0, 0]	[0, 1, 0, 0, 0, 0]
[1, 0, 0, 0, 0, 0]	[0, 0, 1, 0, 0, 0]
[0, 1, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 0]
[0, 1, 0, 0, 0, 0]	[0, 0, 1, 0, 0, 0]
[0, 0, 1, 0, 0, 0]	[1, 0, 0, 0, 0, 0]
[0, 0, 1, 0, 0, 0]	[0, 1, 0, 0, 0, 0]
[0, 0, 0, 1, 0, 0]	[0, 0, 0, 0, 1, 0]
[0, 0, 0, 1, 0, 0]	[0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 1, 0]	[0, 0, 0, 1, 0, 0]
[0, 0, 0, 0, 1, 0]	[0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 0, 1]	[0, 0, 0, 1, 0, 0]
[0, 0, 0, 0, 0, 1]	[0, 0, 0, 0, 1, 0]

13.3.2 Word2vec, Text2vec, GloVe, Paragraph2vec

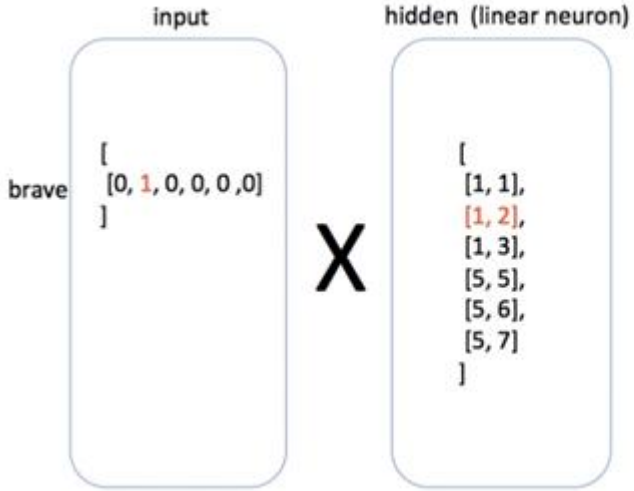
- **Word2vec training**



- **Word2vec is hidden layer after train**



unique word	embedding
king	[1, 1]
brave	[1, 2]
man	[1, 3]
queen	[5, 5]
beautiful	[5, 6]
woman	[5, 7]



unique word	embedding
king	[1, 1]
brave	[1, 2]
man	[1, 3]
queen	[5, 5]
beautiful	[5, 6]
woman	[5, 7]

13.3.2 Word2vec, Text2vec, GloVe, Paragraph2vec

- Word2vec gives similarity in vector representation giving interesting relationships established between the word vectors.

unique word	encoding	word2vec embedding
king	[1, 0, 0, 0, 0, 0]	[1, 1]
man	[0, 0, 0, 1, 0, 0]	[1, 3]
queen	[0, 0, 0, 1, 0, 0]	[5, 5]
woman	[0, 0, 0, 0, 0, 1]	[5, 7]

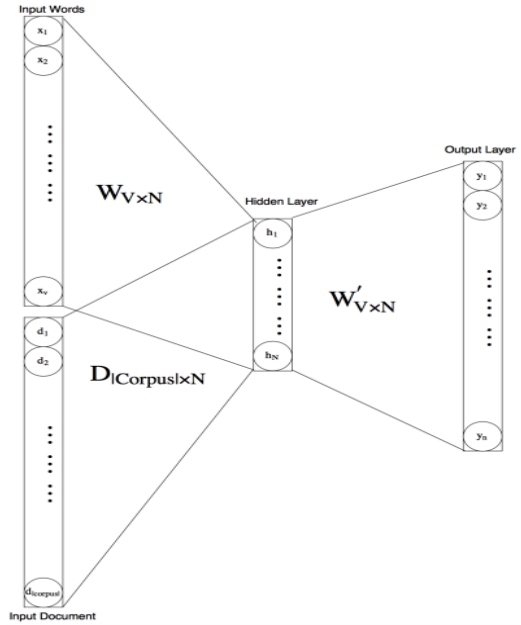
- This means we can also perform vector arithmetic with the word vectors.

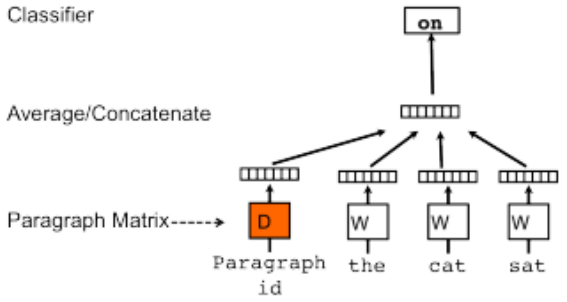
  new_vector = king - man + woman

- This creates new vectors (not directly associated with a word) that we can then attempt to find most similar vectors to.

  new_vector closest to vector for queen

- **Text2vec**
  - Fast text vectorization on arbitrary n-grams
    - using vocabulary
    - using feature hashing
  - State-of-the-art GloVe word embeddings.
- **Paragraph2vec**
  - Every paragraph is mapped into a unique vector D which is a column matrix
  - The only difference from word2vec is inclusion of documents along with words as input nodes.
  - P2V neural net has input nodes representing documents in the training data
  - The rationale behind including documents as input nodes is based upon considering documents as another context.
  - At the time of training pairs(context set, target) are considered as in word2vec, however, for P2V document is also considered a member of the context set.
  - The objective function and the training update steps are exactly the same as word2vec.





Framework of paragraph vector

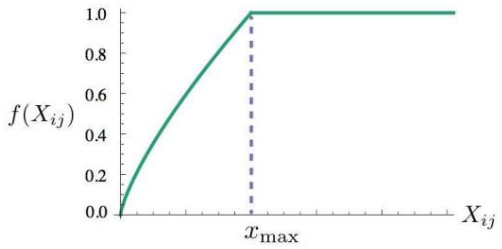
GloVe: Global Vectors for Word Representation

The statistics of word occurrences in a corpus is the primary source of information available to all unsupervised methods for learning word representations.

GloVe combines the advantages of the two major model families in the literature:

- global matrix factorization and,
- local context window method

GloVe Cost function: $J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$



	k=solid	*k=ice*	*k=water*	*k=random*
p(k | ice)	High	Low	High	Low
p(k | steam)	Low	High	High	Low
p(k | ice) / p(k | steam)	< 1	< 1	~ 1	~ 1

Count based versus direct prediction

LSA, HAL, Hellinger-PCA

- Fast training
- Efficient usage of statistics
- Primarily used to capture similarity
- Disproportionate importance given to large words

NLMM, RNN, CBOW-Skipgram

- Scales with corpus size
- Inefficient usage of statistics
- Generate improved performance on other tasks
- Can capture complex patterns

Deriving the Cost Function

- The table beside shows that the appropriate starting point for word vector learning should be with ratios of co-occurrence probabilities rather than the probabilities themselves.

Probability & Ratio	*k=solid*	*k=ice*	*k=water*	*k=random*
p(k | ice)	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
p(k | steam)	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
p(k | ice) / p(k | steam)	8.9	8.5×10^{-2}	1.36	0.96

- Set a function F that represents ratios of co-occurrence probabilities rather than t he probabilities themselves

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}, \qquad (1)$$

→ Note that w_i and $\tilde{w}_k$ are v ectors from **different vector−spaces**

- Encode the information present the ratio in the word vector space. Since vector spaces are inherently linear structures, the most natural way to do this is with vector differences.

$$F(w_i - w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}. \qquad (2)$$

- In equation2, the functions are vectors but the right hand side is a scalar, so take the dot product of the arguments, which prevents F from mixing the vector dimensions in undesirable ways.

$$F\left((w_i - w_j)^T \tilde{w}_k\right) = \frac{P_{ik}}{P_{jk}}, \qquad (3)$$

Deriving the Cost Function

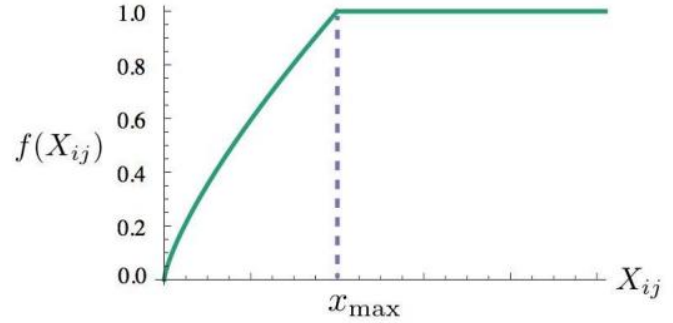
- For word-word co-occurrence matrices, the distinction between a word and a context word is arbitrary and are free to exchange the two roles.
- The symmetry can be restored in two steps:
  - **First**, F be a homomorphism between the groups $(\mathbb{R}, +)$ and $(\mathbb{R}>0, \times)$ $$F\left((w_i - w_j)^T \tilde{w}_k\right) = \frac{F(w_i^T \tilde{w}_k)}{F(w_j^T \tilde{w}_k)}, \qquad (4)$$
  - Using Eqn. 3, Eqn. 4 can be solved by: $$F(w_i^T \tilde{w}_k) = P_{ik} = \frac{X_{ik}}{X_i}. \qquad (5)$$
  - The solution to Eqn. 4 is *exp (exponent)* or $$w_i^T \tilde{w}_k = \log(P_{ik}) = \log(X_{ik}) - \log(X_i). \qquad (6)$$
  - **Second**, Eqn. 6 would exhibit the exchange symmetry if not for the $\mathbf{log}(\boldsymbol{X_i})$ on the right-hand side. This term is however independent of k so it can be absorbed into a bias b_i for w_i. Adding an additional bias $\tilde{b}_i$ for $\tilde{w}_i$ restores the symmetry.

$$w_i^T \tilde{w}_k + b_i + \tilde{b}_k = \log(X_{ik}). \qquad (7)$$

$$f(x) = \begin{cases} (x/x_{\max})^\alpha & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases}. \qquad (9)$$

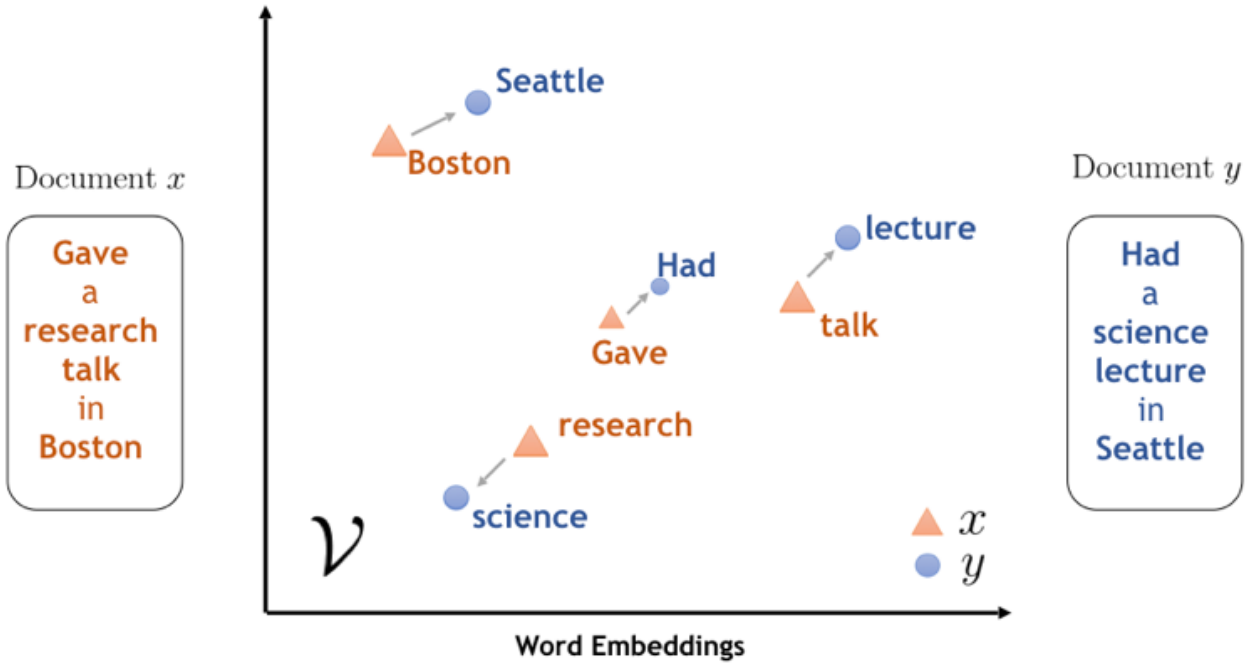
$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2, \qquad (8)$$

and thus we have,



13.3.3 Word Similarity

- Word embedding is the language modeling and feature learning techniques in NLP
- Where words or phrases from the vocabulary are mapped to vectors of real numbers.

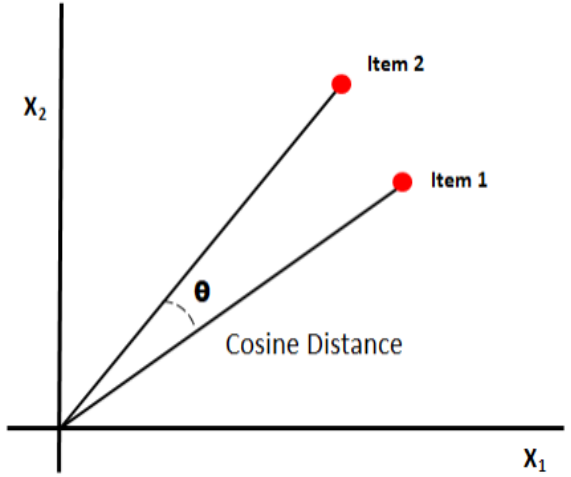


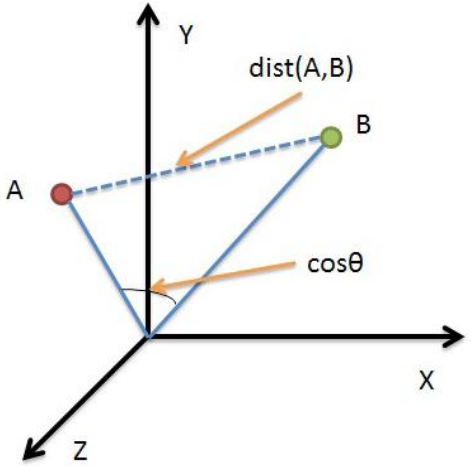
13.3.3 Word Similarity

- In order to find a similarity between words we need to quantify the similarity between words.
- One way of finding the similarity between two words is by edit distance.

Euclidean distance: The distance between two points in the term-document space.

Cosine similarity: By the angle between the vectors; we measure the cosine of the angle. The larger the cosine value, the more similar the documents are or word vectors are to each other.



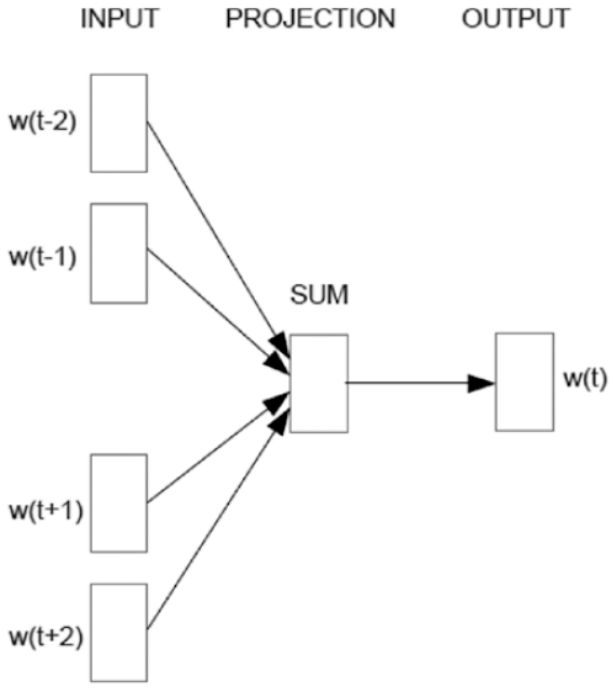


$$Cos(A,B) = \frac{A.B}{|A||B|}$$

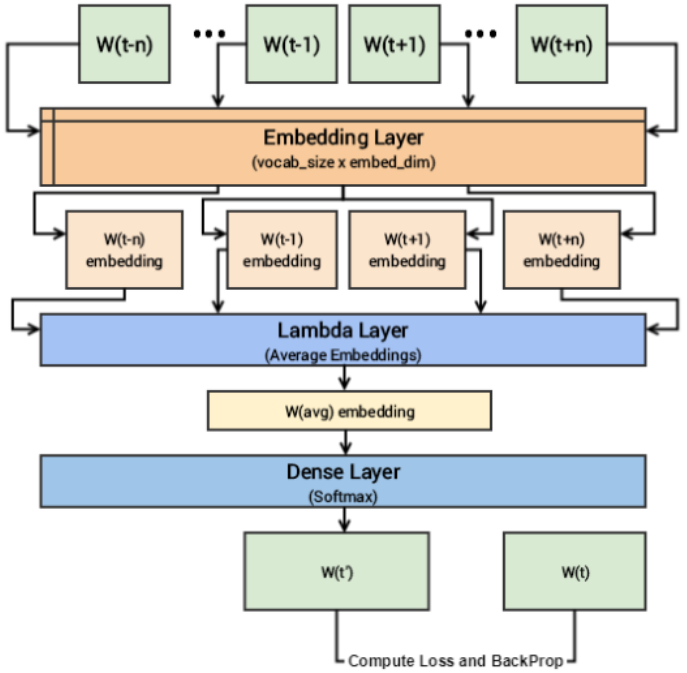
Hamming distance: The number of positions at which the characters are different.

13.3.4 Continuous Bag of Words

- It is one of the models used Word2vec for word embedding
- Continuous bag of words (CBOW) using context to predict a target word based on the surrounding words
- Deep learning architectures are built for designing a CBOW model



Architecture of a CBOW model



Deep learning model for CBOW

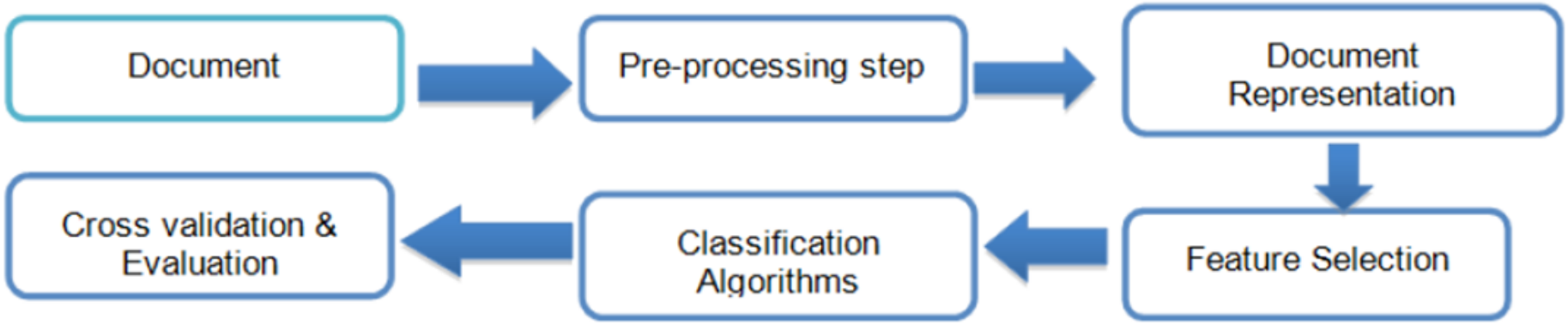
The Topics Covered in This Section

13.4.1 Classification Techniques
13.4.2 Model Evaluation

13.4 Document Classification

- Text classification or text categorization is a supervised learning problem.

Steps:



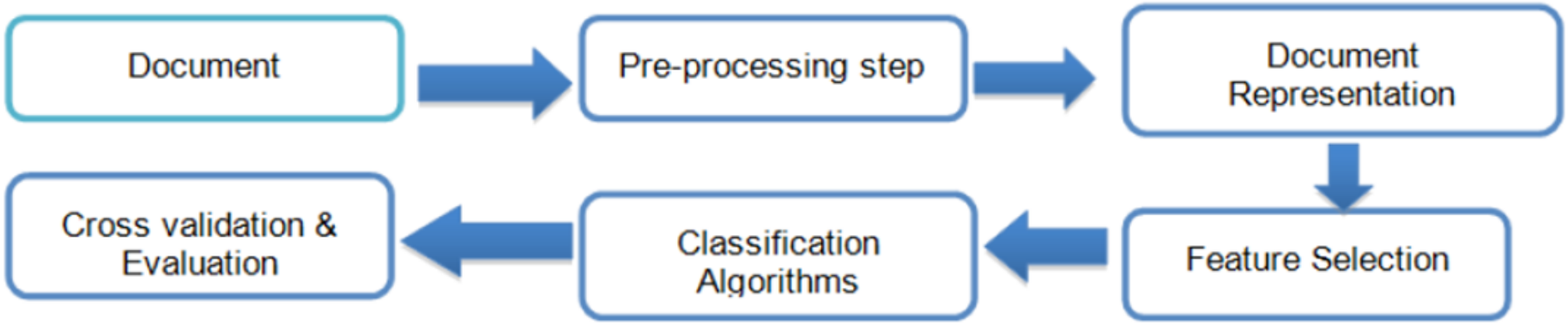
Algorithms:

- Naive Bayes
- KNN
- Support vector machines
- Maximum entropy
- Decision trees

Document Representation

- Representing the document in a manner which is suitable for classification tasks.
- Reduce the complexity of documents, making it easier to work with.

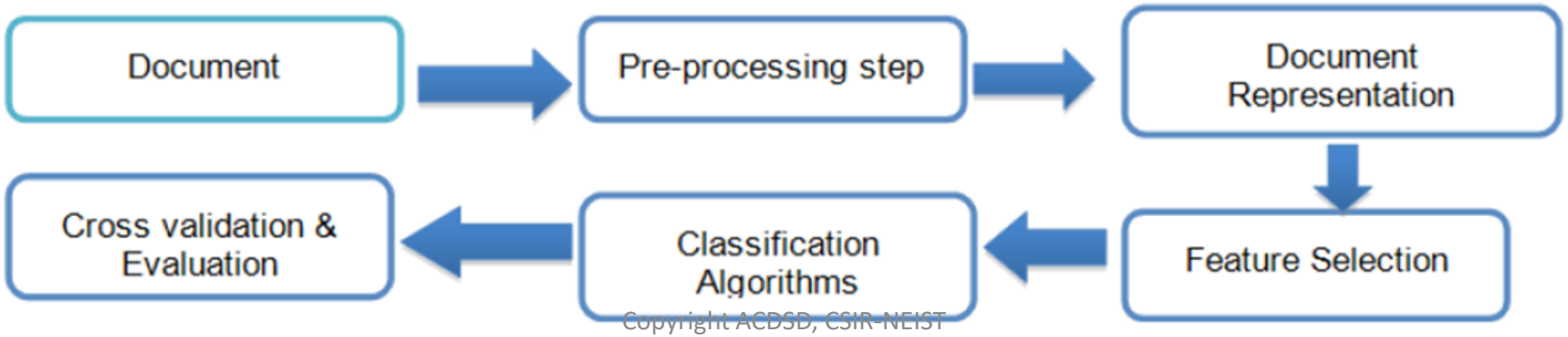
- While doing so, the following questions come to mind:
  - Do we need to preserve the order of words?
  - Is losing the information about the order a concern for us?



Feature Selection

- Each unique word in a document considered as a feature and the frequency of its occurrence is represented as a value.
- Thus Tf and Tf-Idf are also considered as features of a document.
- Discarding the features with a very low occurrence can reduce the high dimensionality.

- Vector space representation of words considers each word in a document as a vector.
- Vector space models have some limitations too.
- For example, the document representation is very high dimensionally, and it may also lead to the loss of semantic relations or correlations between the different terms in the given documents.



13.4.2 Classification Model Evaluation

- Confusion Matrix
  - Accuracy
  - Precision
  - Recall
  - F1 score
- ROC curve
- Bias variance Trade off
- Cross Validation
  - Leave-one-out
  - k-fold
  - Repeated hold
  - Bootstrap
  - Stratified

		predicted condition	
	total population	prediction positive	prediction negative
true condition	condition positive	**True Positive (TP)**	**False Negative (FN)** (type II error)
	condition negative	**False Positive (FP)** (Type I error)	**True Negative (TN)**

The Topics Covered in This Section

13.5.1 Clustering Techniques
13.5.2 Model Evaluation

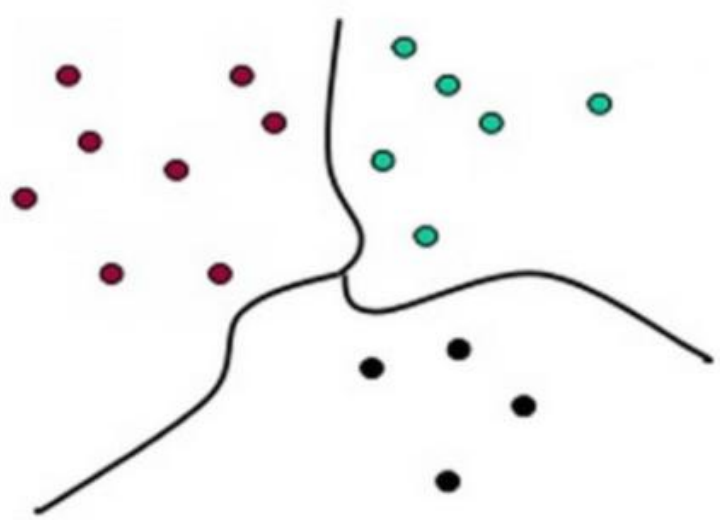
13.5 Document Clustering

- Text clustering is an unsupervised learning that helps to find and group similar objects together.
- Text clustering is utilized to group text objects of different granularities such as documents, paragraphs, sentences, or terms together.

- The hypothesis of the clustering algorithm is based on minimizing the distance between objects in a cluster, while keeping the intra-cluster distance at maximum.

Algorithms:

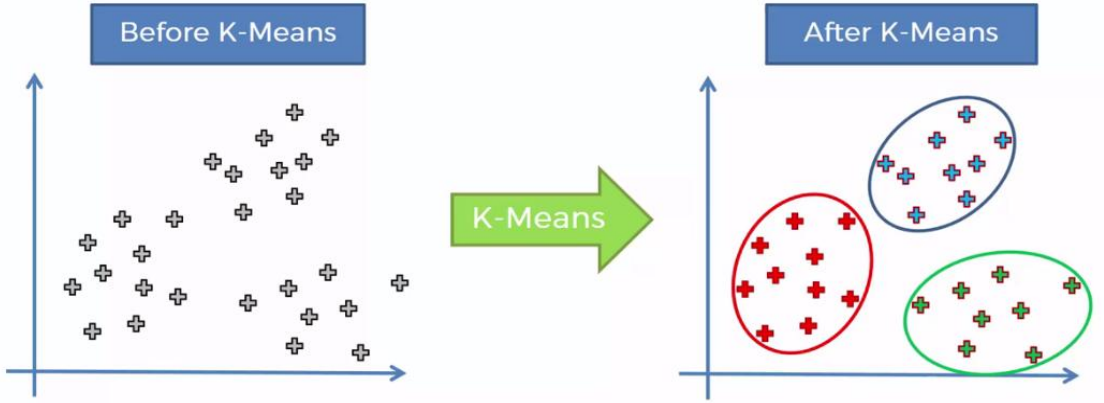
- K-means
- Hierarchical clustering



13.5.1 Clustering Techniques

K-means

- Clustering algorithm for partitioning a given data set into a set of k groups (i.e. *k clusters*), where k represents the number of groups pre-specified by the analyst.
- K-means algorithm allows that each one of the data belong to only a cluster.
- Therefore, this algorithm is a definite clustering and hard clustering algorithm.



Divides the observations into discrete groups based on some distance metric

13.5.1 Clustering Techniques

ALGORITHM: K-Means

Input: $\{x^1, x^2, x^3 ..., x^n\}$ //Training Set

K // Number of desired clusters

Output: K // Set of Clusters

Randomly initialize K cluster centroids as $\{\mu_1, \mu_2, \mu_3 ... \mu_K\}$

Repeat

for i=1 to n

$c^{(i)}$ = index of cluster centroid (from 1 to K) closest to $x^{(i)}$

for k=1 to K

μ_K = average(mean) of points assigned to cluster k

Until convergence criteria is meet

13.5.1 Clustering Techniques

- Optimization objective: $J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K) = \frac{1}{m} \sum_{i=1}^{m} ||x^{(i)} - \mu_{c^{(i)}}||^2$

  where,

  - μ_k is the cluster associated with centroid *k*
  - c^i is the index of clusters {1,2, ..., K} to which x^i is currently assigned.
  - So, $\mu_c{}^i$ is the cluster centroid of the cluster to which example x^i has been assigned to.

- Minimizing cost: $\min_{\substack{c^{(1)},\ldots,c^{(m)},\\ \mu_1,\ldots,\mu_K}} J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K)$

  where,

  - The **cluster assigned step** is minimizing J(...) with respect to c^1, c^2 ... c^i
  - The **move centroid step** is minimizing J(...) with respect to μ

Initializing the centroids – Random Initialization

- Randomly pick K training examples
- Set μ_1 up to μ_K to these example's values

Effects of Random Initialization

- Kmeans can converge to different solutions depending on the initialization setup
- That means risk of local optimum.
- The local optimum are valid convergence, but local optimum are not global ones
- If this is a concern go for multiple random initializations.

Multiple Random Initialization:

for i=1 to 100 {

Randomly initialize K-means;

Run K-means;

Get $c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K$

Compute Cost Function (distortion) $J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K)$

}

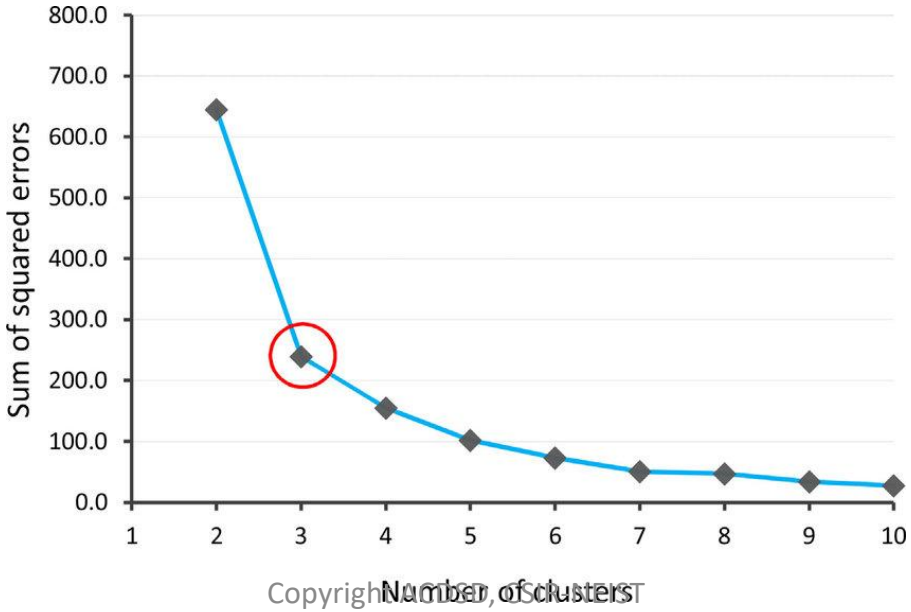
Pick the lowest distortion clusters.

Deciding K - Elbow method

- Compute cost function at a range of K values (As K increases J(...) minimum value decreases)
- Plot (K vs J())
- Look for the "elbow" on the graph and Chose the "elbow" number of clusters.

Risks:

- Normally we don't get a nice line – no clear elbow on curve
- Not really that helpful



13.5.1 Clustering Techniques

Strength:

- Easy to implement.
- Efficient: O(tkn) where n is number of objects, k is number of clusters, and t is number of iterations.
- k-Means produce tighter clusters than hierarchical clustering.
- K-means is scalable and efficient for large dataset with complexity O(nkt).

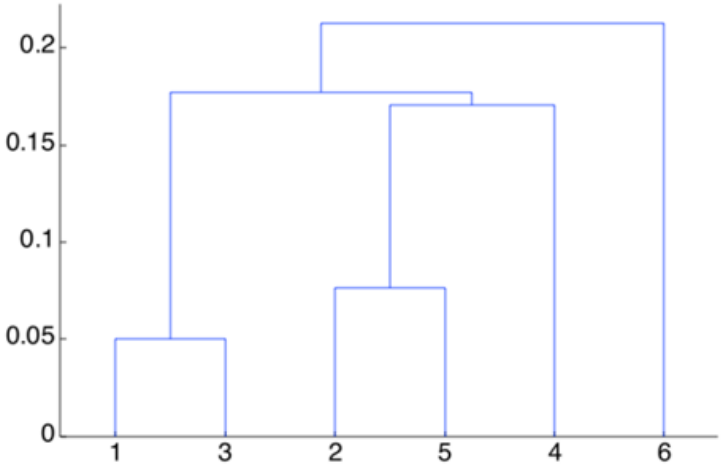
Weakness:

- Need to specify k, the number of clusters, in advance.
- Initial seeds have a strong impact on the final results.
- Applicable only when mean is defined - Categorical data.
- Unable to handle noisy data and outliers.
- Not suitable to discover clusters with non-convex shapes.

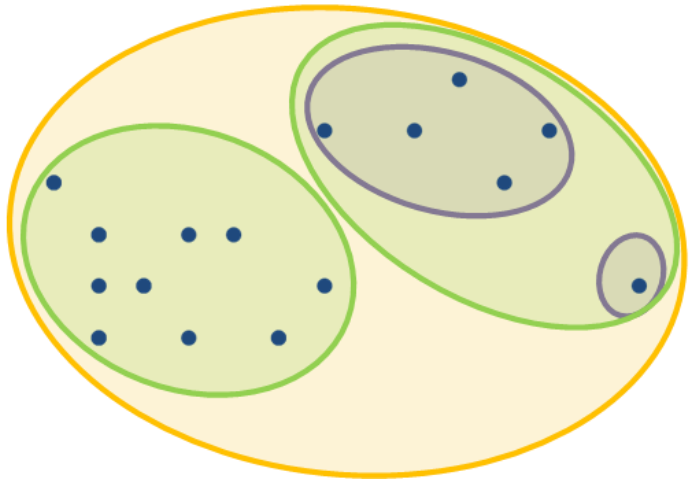
13.5.1 Clustering Techniques

Hierarchical Clustering

- Produces a set of nested clusters organized as a hierarchical tree.
- Can be visualized as a Dendogram.
- A tree like diagram that records the sequences of merges or splits

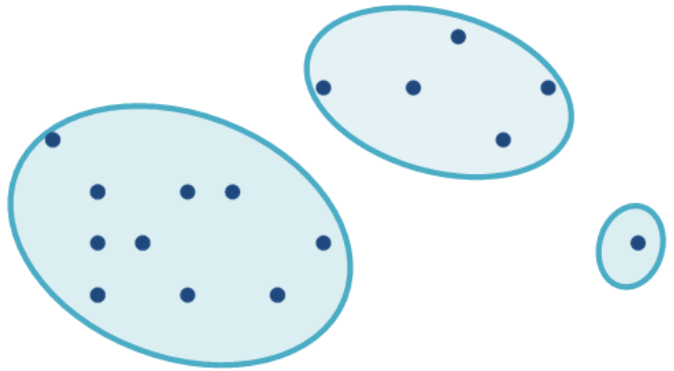


Hierarchical Clustering

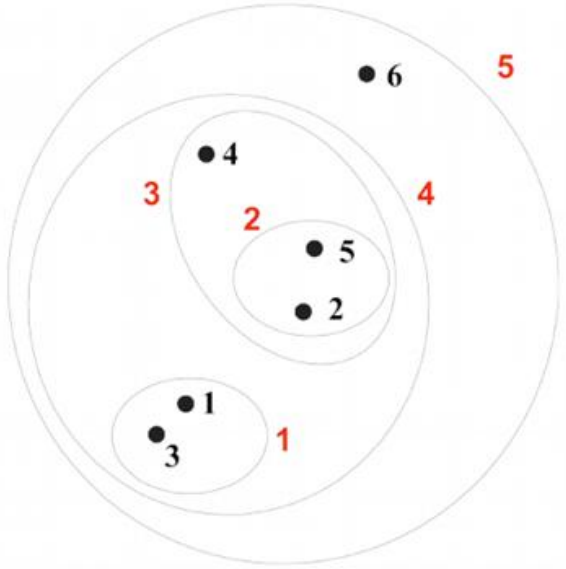


Creates a nested and hierarchical set of clusters

Partitional Clustering



Each sample(point) is assigned to a unique cluster



13.5.1 Clustering Techniques

Agglomerative (Bottom Up):

- Start with considering each point as individual cluster (leaf).
- At each step, two clusters that are the most similar are combined into a new bigger cluster (nodes).
- This procedure is iterated until all points are member of just one single big cluster (root).

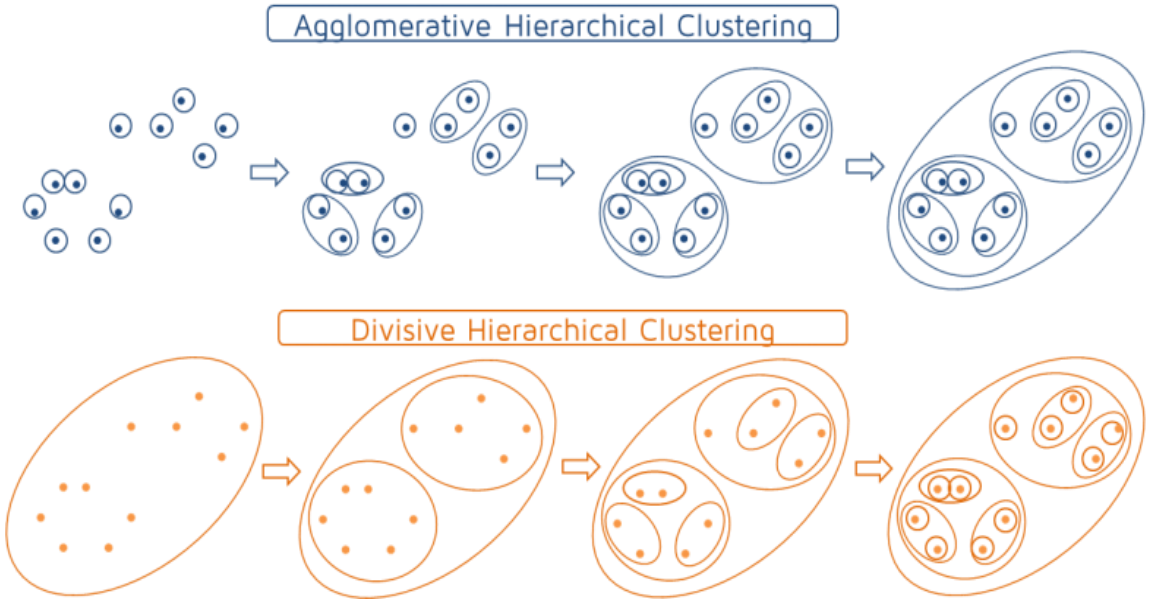
Divisive (Top Down):

- Start with considering all points as one single cluster (root).
- At each step of iteration, the most heterogeneous cluster is divided into two.
- The process is iterated until all objects are in their own cluster.

13.5.1 Clustering Techniques

AGNES vs DIANA

- AGNES (Agglomerative Nesting) clustering is good at identifying small clusters.
- DIANA (Divise analysis) clustering is good at identifying large clusters.



13.5.2 Model Evaluation

- **Direct Methods:** Consists of optimizing a criterion.
  - Elbow method based on within cluster sums of squares
  - Silhouette method based on average silhouette.

- **Statistical Testing Methods:** Consists of comparing evidence against null hypothesis.
  - Gap Statistic

The Topics Covered in This Section

13.6.1 Topic Modeling vs Clustering
13.6.2 Latent Dirichlet Allocation (LDA)
13.6.3 Probabilistic Latent Semantic Indexing (pLSI)
13.6.4 Variations

13.6.1 Topic Modeling vs Clustering

- Topic modeling is a probabilistic model to extract the underlying themes or topics (latent features) that are present in a collection of document.
- It helps summarize a document or organize a group of them based on the discovered topics, so that users can easily browse through the documents based on topics of their interest.

Clustering vs Topic Modelling:

- Clustering documents results in each text exclusively belong to exactly one cluster.
- Let's consider this scenario: The pdf "*Text Mining and Natural Language Processing*" should be grouped with Text Mining and NLP both not only one group.
- Topic modeling do not cluster documents into completely separate groups.

Techniques:

- Latent Dirichlet Allocation (LDA)
- Correlated Topic Model (CTM)
- Non-negative Matrix Factorization (NMF)

13.6.2 Latent Dirichlet Allocation

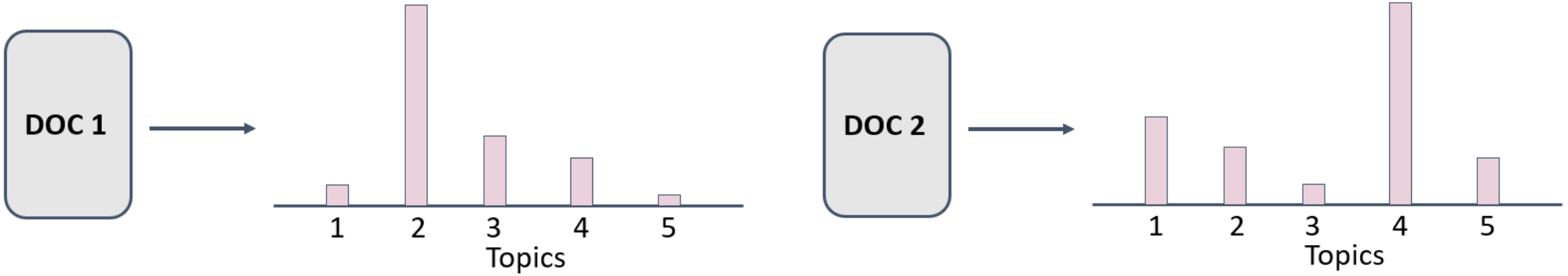
- LDA is based on “Dirichlet Distribution” by Johann Peter Gustav Lejeune Dirichlet.
- In 2003 LDA was first published as a graphical model for topic discovery in *Journal of Machine Learning Research* by David Blei, Andrew Ng and Michael I. Jordan.

Assumptions of LDA for Topic Modeling

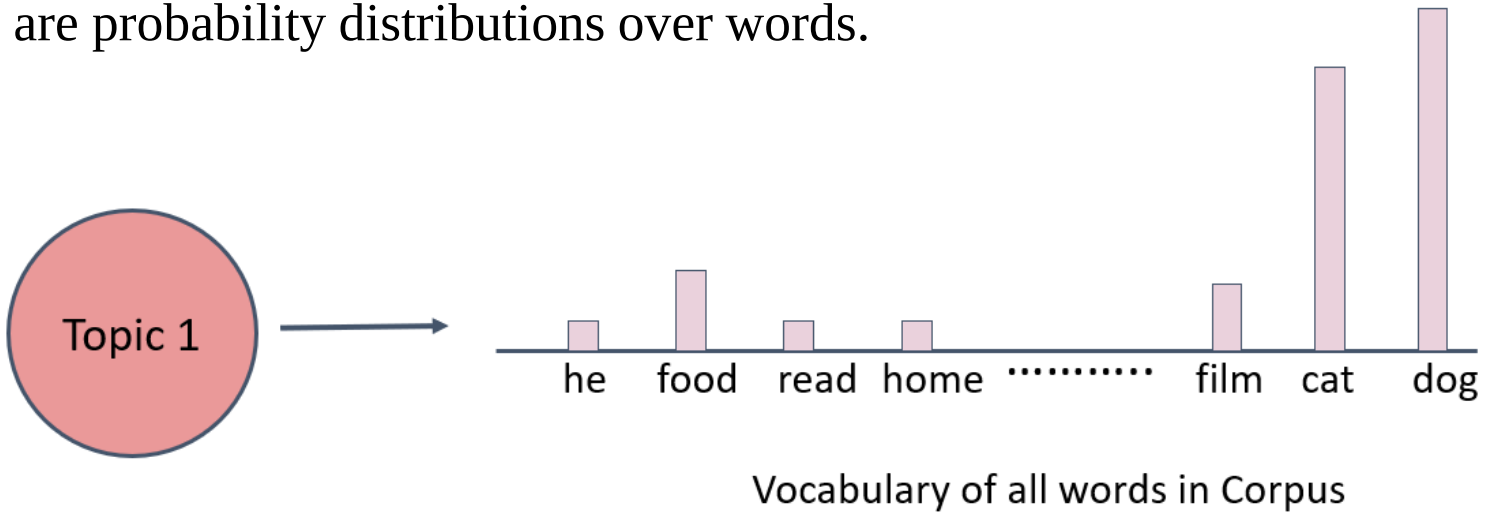
- Documents are probability distributions over latent topics.
- Topics themselves are probability distributions over words.
- Documents with similar topics use similar groups of words.
- Latent topics can then be found by searching for groups of words that frequently occur together in documents across the corpus.
- The order of the documents and words does not matter as it will treat each document as a BoW.
- This assumption may not be the best, since the sequence of the words in the sentence are lost.
- It helps us knowing which words were used in a document and their frequencies makes it good enough to make decisions on which topic they belong to.

13.6.2 Latent Dirichlet Allocation

- Documents are probability distributions over latent topics.



- Topics themselves are probability distributions over words.



13.6.2 Latent Dirichlet Allocation

Initial Decisions to make: (considering documents are produced in the following fashion)

- Decide on the number of words N the document will have.
- Choose a topic mixture for the document over a fixed set of K topics (a/t Dirichlet distribution).
- For example, 60% business, 20% politics, 10% food.

Generate each word in the document:

- Pick a topic according to topic's multinomial distribution (60% business, 20% politics, 10% food)
- Using the topic to generate the word itself (according to the topic's multinomial distribution).
- For example, if we selected the food topic, we might generate the word "apple" with 60% probability, "home" with 30% probability, and so on.

Generate each topic in the document:

- Assuming this generative model for a collection of documents, LDA then tries to backtrack from the documents to find a set of topics that are likely to have generated the collection.

13.6.2 Latent Dirichlet Allocation

- Now imagine we have a set of documents.
- We've chosen some fixed number of K topics to discover, and want to use LDA to learn the topic representation of each document and the words associated to each topic.

- Go through each document and randomly assign each word in the document to one of the K topics.
- This random assignment already gives you both topic representations of all the documents and word distributions of all the topics (note, these initial random topics won't make sense).

- Now we iterate over every word in every document to improve these topics.
  - For every word in every document and for each topic **t** we calculate:
    p(topic t | document d) and p(word w | topic t)
  - Reassign w a new topic, where we choose topic t with probability
    p(topic t | document d) * p(word w | topic t)

13.6.2 Latent Dirichlet Allocation

- After repeating the previous step a large number of times, we eventually reach a roughly steady state where the assignments are acceptable.
- At the end we have each document assigned to a topic.
- We also can search for the words that have the highest probability of being assigned to a topic.

We end up with an output such as:

- Document assigned to Topic #4
- Most common words (highest probability) for Topic #4: [‘cat’,’vet’,’birds’,’dog’,...,’food’,’home’]
- It is up to the user to interpret these topics.

Notes:

- The user must decide on the amount of topics present in the document.
- The user must interpret what the topics are.

13.6.3 Probabilistic Latent Semantic Indexing

Latent Semantic Indexing: Perform a low-rank approximation of document-term matrix (typical rank 100-300)

General Idea:

• Map documents (and terms) to a low-dimensional representation (PCA).

• Design a mapping such that the low-dimensional space reflects semantic associations (latent semantic space).

• Compute document similarity based on the inner product in the latent semantic space.

Some Assumptions:

• We have a set of documents $dd_1, dd_2, \ldots, dd_N$.

• Each document is just a collection of words or a "bag of words". Thus, the order of the words and the grammatical role of the words (subject, object, verbs, …) are not considered in the model.

• Words like am/is/are/of/a/the/but/… (stop words) can be eliminated from the documents as a preprocessing step since they don't carry any information about the "topics".

• In fact, we can eliminate words that occur in at least %80 ~ %90 of the documents!

13.6.3 Probabilistic Latent Semantic Indexing

Singular Value Decomposition (SVD):

$$A = U\Sigma V' \in \mathbb{R}^{n\times m}$$

$$U \in \mathbb{R}^{n\times k} \qquad \Sigma \in \mathbb{R}^{k\times k} \qquad V \in \mathbb{R}^{m\times k}$$

$$U'U = I \qquad V'V = I \qquad \Sigma = diag(\sigma_1, \dots, \sigma_k), \sigma_i \geq \sigma_{i+1} \qquad k = rank(A)$$

Approximation Problem:

$$X^* = \underset{\hat{X}:rank(\hat{X})=q}{\operatorname{argmin}} \left\|X - \hat{X}\right\|_F, \qquad Ferbenius\ Norm\ \|A\|_F \stackrel{\text{def}}{=} \sqrt{\sum\sum |a_{ij}|^2}$$

PLSI Model Definition

Vocabulary Mismatch Problem:

- One concept can be represented by several different words!
- Two documents might not contain similar terms (for instance due to writing styles) but refer to a single concept.
- Queries can contain words not present in a document and still be very relevant to that document!

We look for P(a word or a query| the context)

$$P(R_d = 1|q) = \frac{P(q|R_d = 1)P(R_d = 1)}{P(q)} \propto \underbrace{P(q|R_d = 1)}_{\text{Given a document how probable is a query}} \quad \underbrace{P(R_d = 1)}_{\text{Uniform or relevant to the popularity of the document}}$$

$R_d \in \{0,1\}$: relevance of a document
q: a query, set of words

13.6.3 Probabilistic Latent Semantic Indexing

Calculating $P(q|R_d = 1)$:

• For each document calculate the probability of each word w coming from (or being relevant to) that document i.e. $P(w|\ R_d = 1)$

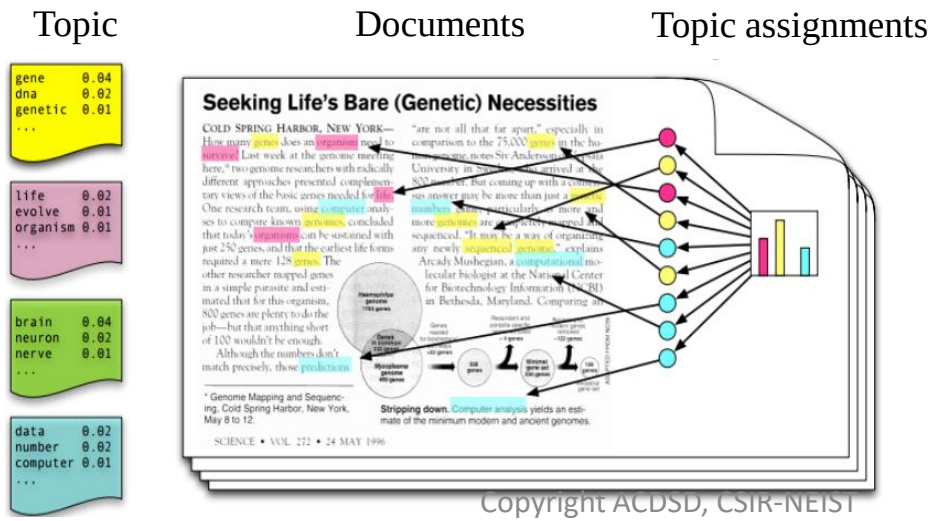
• Calculate the conditional probability of the words in q

PLSI Model Elements:

• A set of documents $\{d_1, \dots, d_N\}$

• A set of concepts, classes or topics $\{z_1, \dots, z_K\}$

• A set of words $\{w_1, \dots, w_M\}$



- Each concept is a distribution over words
- Each document is a mixture of corpus-wide topics
- Each word is drawn from one of these topics
- We only observe the words within the documents and the other structures are hidden variables.

Correlated Topic Modeling

- CTM is a extension of LDA, building upon the LDA model.
- Gives better predictive performance, but comes at the expense of extra computation cost.

- LDA model does not take into account the order in which the words occur.
- CTM Helps to understand the correlation between the hidden topics in the collection of documents.

The Topics Covered in This Section

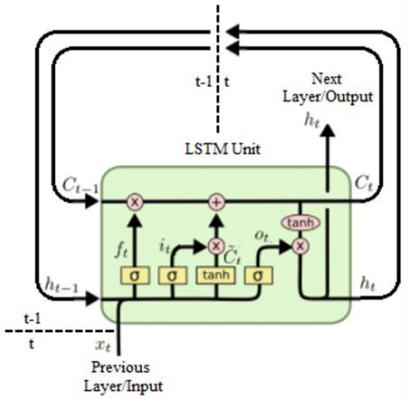
13.7.1 Text Generation with LSTM
13.7.2 Creating NER
13.7.3 Sequence to Sequence Model
13.7.4 Question and Answering Model

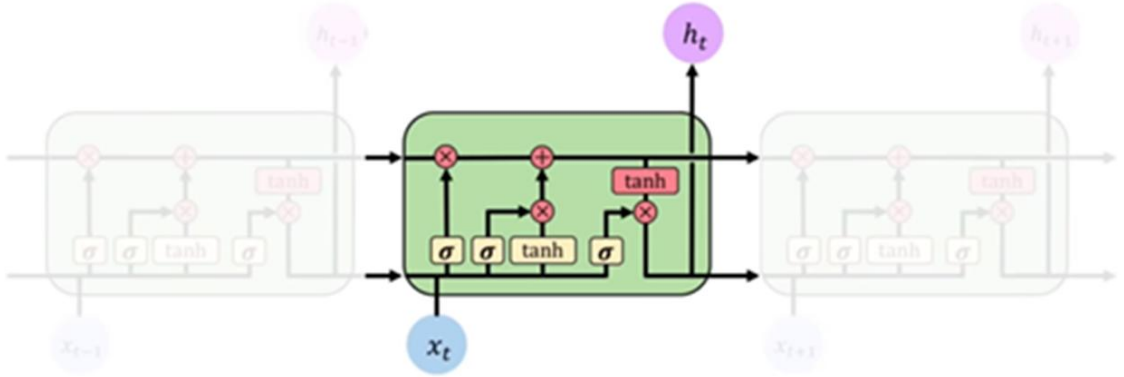
13.7.1 Text Generation with LSTM

- Generate New Text Based off a Seed

How?

- Create the LSTM Based Model
- Split the Data into Features and Labels
  - X Features (First n words of Sequence)
  - Y Label (Next Word after the sequence)
- Fit the Model





What is Corpus?

- Text corpus (corpora) is a bundle of large and structured set of texts.
- Prepared from all the collected text data for further processing and analysis.

What's not ?

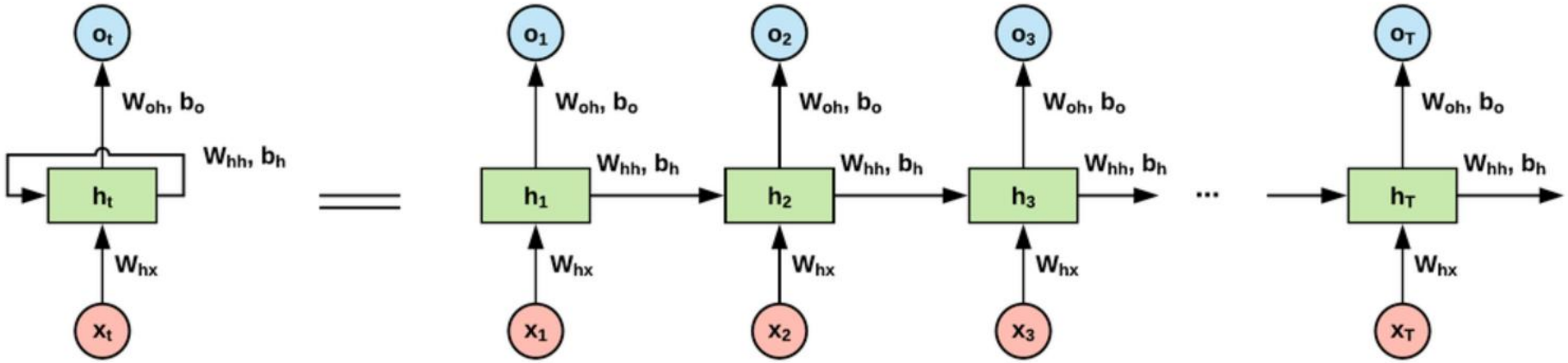
- Not a database or text archive.
- Not necessary in ordered.

Creating NER

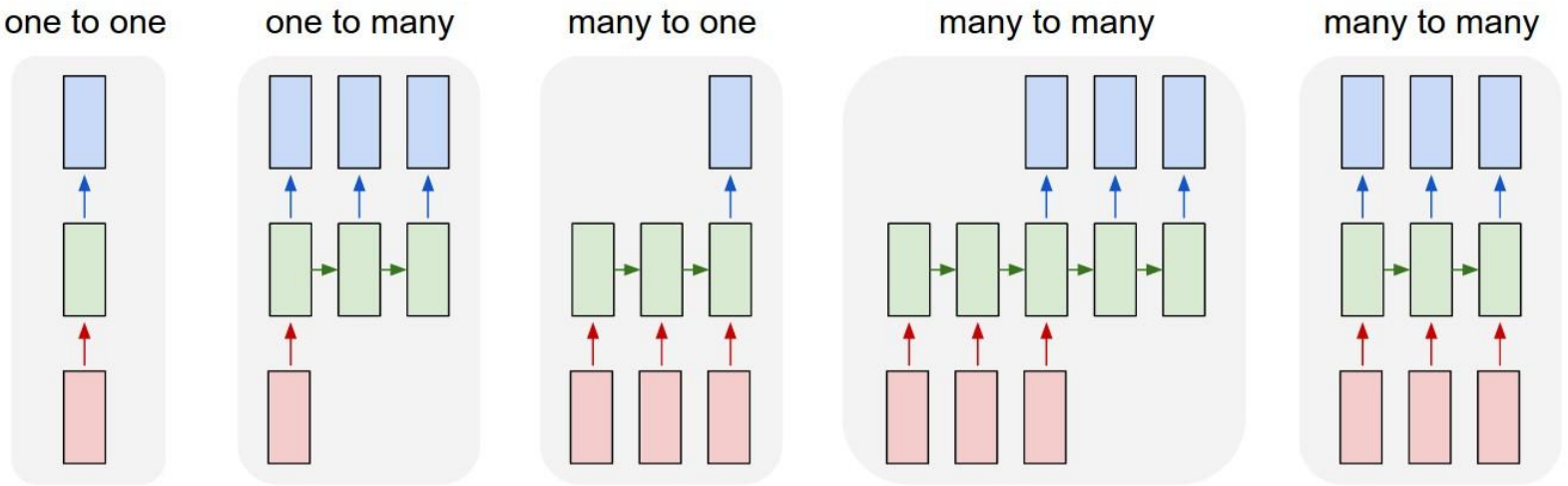
- Rule based method: Using the linguistic rules for recognizing named entities
- Statistical method: Using a corpus to train a model like HMM, CRF etc. and then identify the entities
- Hybrid: Combination of both rule based and statistical method for optimum results

13.7.3 Sequence to Sequence Model

Recurrent Neural Networks (RNN)

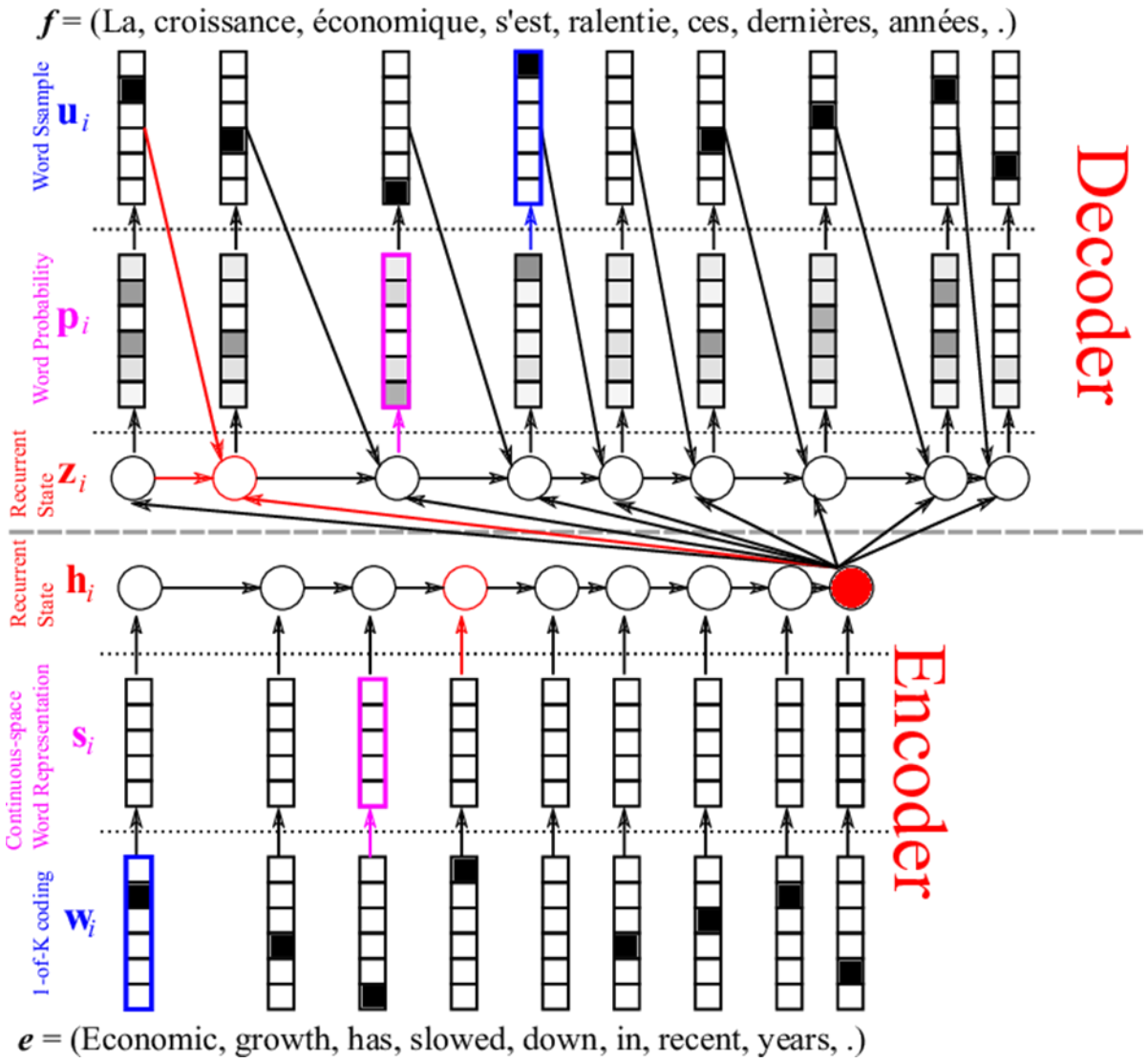


$$h_t = g\,(w_{xh}x_t + w_{hh}h_{t-1} + b_h)$$
$$o_t = g\,(w_{oh}h_t + b_o)$$



13.7.3 Sequence to Sequence Model

- Sequence-to-sequence learning (Seq2Seq) is about training models to convert sequences from one domain (e.g. English) to sequences in another domain (e.g. translated to French)
- A typical sequence to sequence model has two parts – an ***encoder*** and a ***decoder.***
- Both the parts are practically two different neural network models combined into one giant network.
  - A RNN layer acts as "encoder“ to processes the input sequence and returns its own internal state
  - Another RNN layer acts as "decoder“ which is trained to predict the next characters of the target sequence, given previous characters of the target sequence.



13.7.4 Question Answering Model

- A chat bot that can answer questions based on a “story” given to the bot.

- **Story:** Jane went to the store. Mike ran to the bedroom.
- **Question:** Is Mike in the store?
- **Answer:** No

End-to-End Memory Networks [Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, Rob Fergus]

- Model takes a discrete set of inputs **x1, ..., xn** that are to be stored in the memory, a query **q**, and outputs an answer **a**
- Each of the **x** , **q**, and **a** contains symbols coming from a dictionary with **V** words.
- The model writes all **x** to the memory up to a fixed buffer size, and then finds a continuous representation for the **x** and **q**.

13.7.4 Question Answering Model

End-to-End Memory Networks

- Input Memory Representation of stories.

$$p_i = \text{Softmax}(u^T m_i)$$

$$\text{where Softmax}(z_i) = e^{z_i} / \sum_j e^{z_j}.$$

- Embedding sentences **x**
- Encoders C and M
- Question Encoder

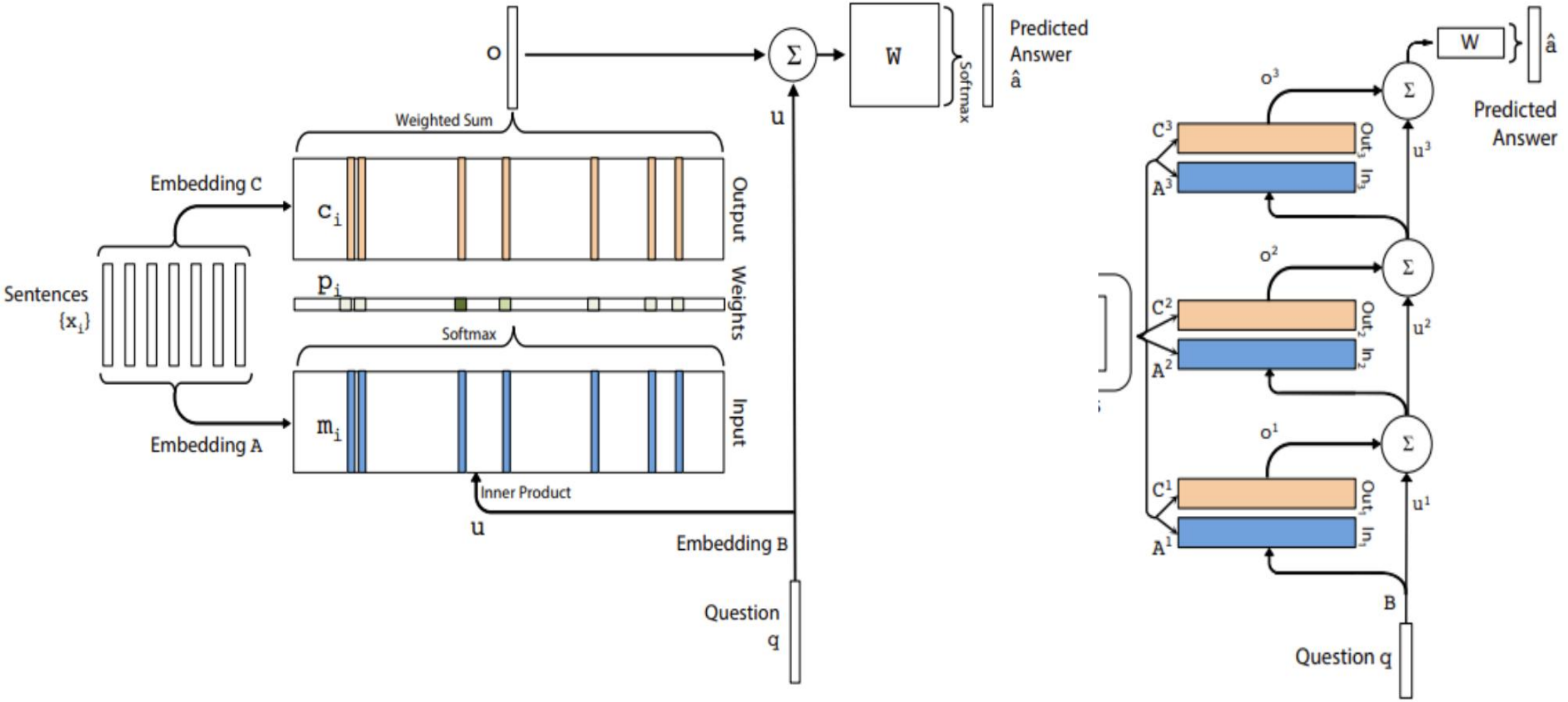
- Output Memory Representation

$$o = \sum_i p_i c_i.$$

- Each **x** has a corresponding output vector
- In the single layer case, the sum of the output vector **o** and the input embedding u is then passed through a final weight matrix W (of size V × d) and a softmax to produce the predicted label:

$$\hat{a} = \text{Softmax}(W(o + u))$$

End-to-End Memory Networks



13.8 Implementation

- 13.8.1 R Text Mining Packages
- 13.8.2 Datasets and Loading Data
- 13.8.3 View File
- 13.8.4 Text Preprocessing
- 13.8.5 Document Classification
- 13.8.6 Document Clustering
- 13.8.7 Topic Modelling
- 13.8.8 Text Generation

13.8.1 R Text Mining Packages

To install a R package:

install.packages('package name')

Visualization

wordcloud
ggplot2
SnowBallC
textplot
zipfR

Framework

tm (core)
openNLP
Rweka
KoRpus
tokenizers
textreuse
stm
gsubfn
tidyverse
tidytext

Text Analytics

tm
openNLP
Rweka
KoRpus
lda
lsa
topicmodels
skmeans
textcat
wordnet
corpora
text2vec
word2vec

13.8.2 Datasets and Loading Data

- Text Mining works largely on datasets which can be locally built or exported from different sources
- Datasets are also popularly known as “corpus”
- To build a corpus, save the raw data into a file which may be of various extensions like txt, csv, xls etc. in a proper location.

Loading data from a dataset

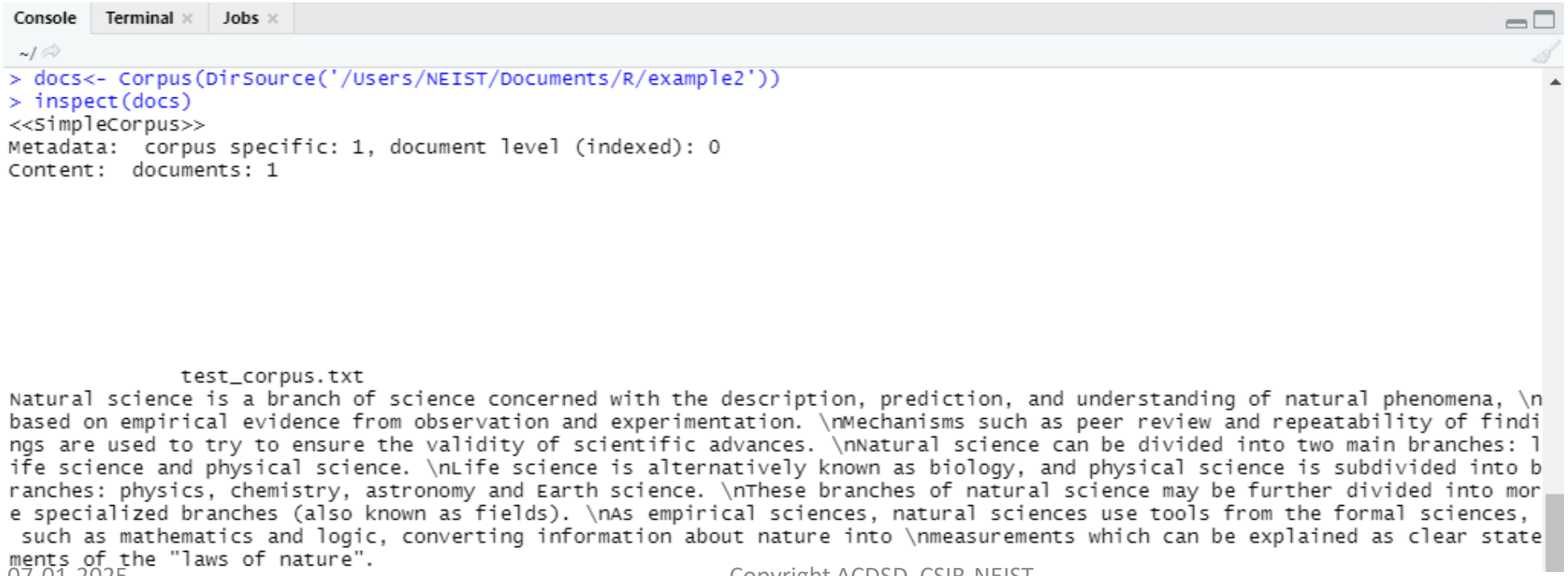
```
install.packages('tm')
library(tm)
docs<- Corpus(DirSource('/Users/NEIST/Documents/R/example2'))
```

The dataset available at the specified location is loaded into corpus ***docs.***

13.8.3 View File


```
install.packages('tm')
library(tm)
docs<- Corpus(DirSource('/Users/NEIST/Documents/R/example2'))
inspect(docs)        // docs is the corpus of raw data and using 'inspect' the corpus is being read
```

Output:



13.8.4 Text Preprocessing

Loading of data:

```
docs<- Corpus(DirSource('/Users/NEIST/Documents/R/example3'))
```

Preprocessing:

```
// create a content transformer to modify the content of an R object
toSpace <- content_transformer(function(x, pattern){return(gsub(pattern, " ",x ))})

docs <- tm_map(docs,toSpace,"-")
docs <- tm_map(docs,toSpace,":")
docs <- tm_map(docs,toSpace,",")
docs <- tm_map(docs,toSpace,"'")
docs <- tm_map(docs,toSpace," _")
docs <- tm_map(docs,toSpace,"\n")
docs <- tm_map(docs, removePunctuation)
docs <- tm_map(docs, removeNumbers)
docs <- tm_map(docs, content_transformer(tolower))
docs <- tm_map(docs, removeWords, stopwords("english"))
docs <- tm_map(docs, stripWhitespace)
```

13.8.5 Document Classification

- Automatically assign a document to one or more classes
- Documents may be classified according to their subjects or according to other attributes
- Automatically classify unlabeled documents to a set of relevant classes using labeled training data

Sentiment Analysis for Movie Reviews

Scenario: A movie website allows users to submit reviews describing what they either liked or disliked about a particular movie.
Problem: The user reviews are unstructured text. How to automatically generate a score indicating whether the review was positive or negative?
Solution: Train a natural language parsing model on a dataset that has been labeled in previous reviews as either positive or negative.

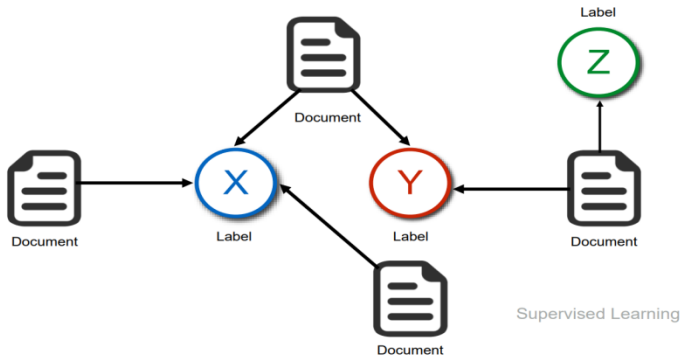
Recommend relevant tags

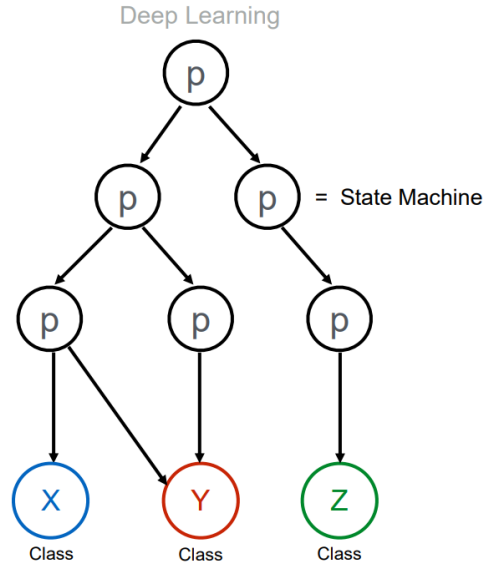
Scenario: A Q/A website allows users to submit questions and receive answers from other users.
Problem: Users sometime do not know what tags to apply to their questions in order to increase discoverability for receiving answers.
Solution: Automatically recommend the most relevant tags for questions by classifying the text from training on previous questions.

Recommend relevant articles

Scenario: A news website provides hundreds of new articles a day to users on a broad range of topics.
Problem: The site needs to increase user engagement and time spent on the site.
Solution: Train natural language parsing models for daily articles in order to provide recommendations for highly relevant articles at the bottom of each page.

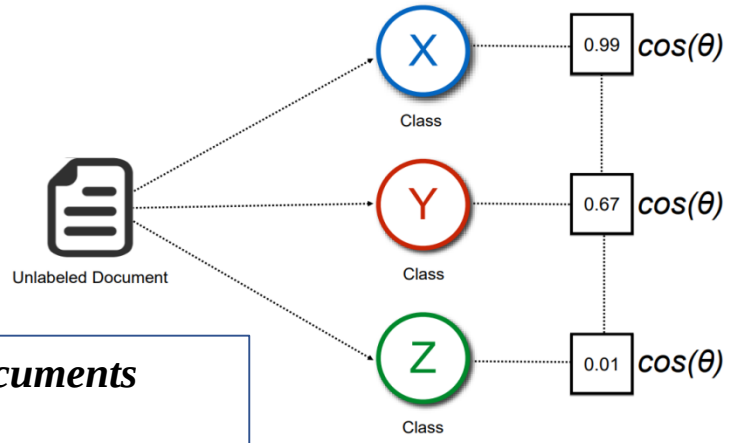
13.8.5 Document Classification





Creating a training dataset

Assign a set of labels that describes the document's text



Training NLP parsing model

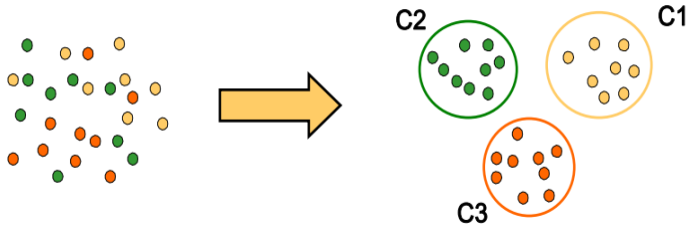
- ❑ Deep feature representations are selected and learned using an evolutionary algorithm
- ❑ State machines represent predicates that evaluate to 0 or 1 for a text match
- ❑ State machines map to classes of document labels that matched text during training

Classify unlabelled documents

The NL parsing model is used to classify other unlabeled documents

13.8.6 Document Clustering

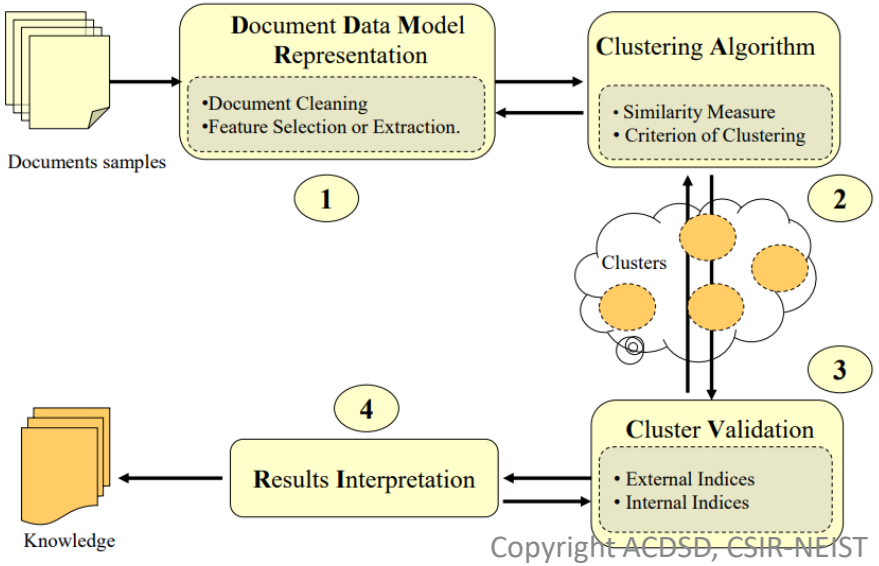
Document clustering is an automatic grouping of text documents into clusters so that documents within a cluster have high similarity in comparison to one another, but are dissimilar to documents in other clusters.

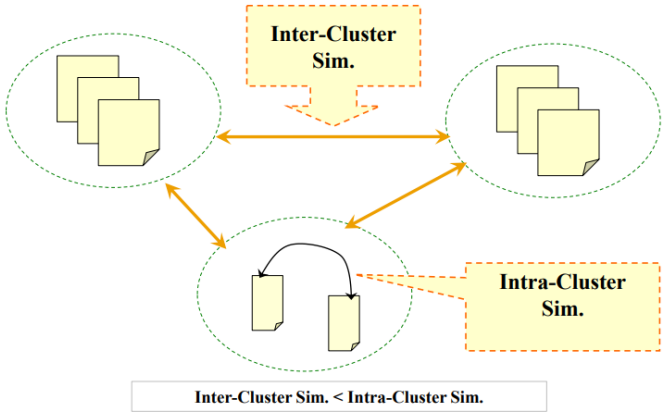


The problem of Document clustering is how to organize a large set of documents of various topics and reach satisfy organization.

Given: A huge set of documents of various topics (shared, related, totally different).
Required: Group the documents into a number of clusters such that the intra-cluster similarity is maximized, and the inter-cluster similarity is minimized.

Document clustering process

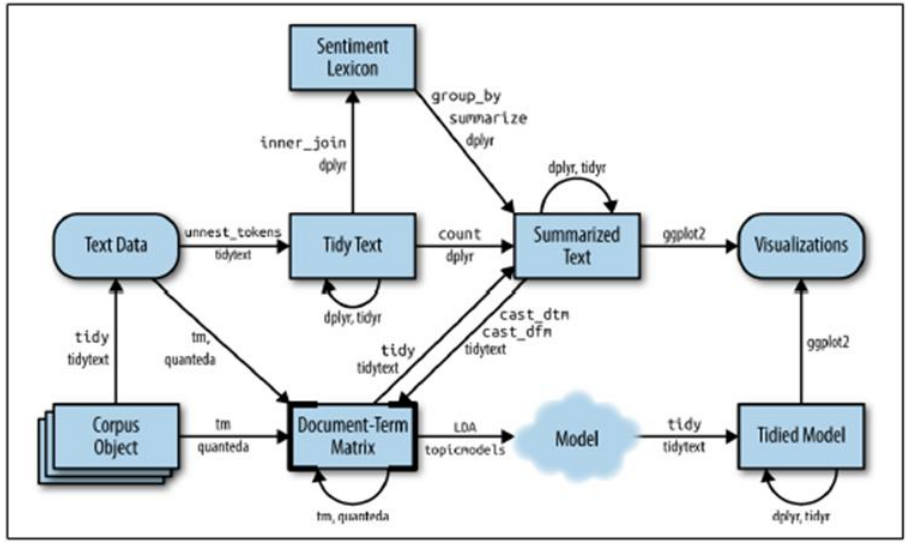


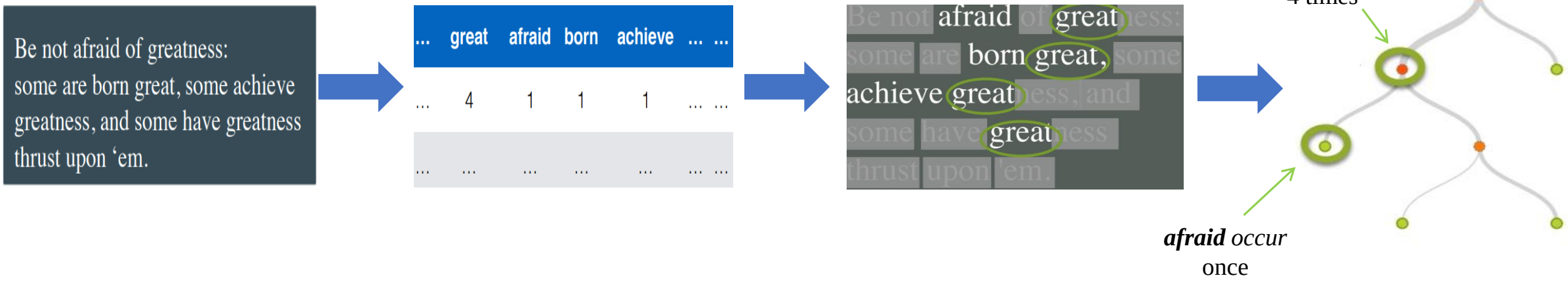


13.8.7 Topic Modeling

- Learns from text fields and finds the hidden topics that model the text
- Finds topics in your text fields
  - A topic is a distribution over terms
  - Terms with high probability in the same topic often occur together in the same document
- Steps for implementing topic modeling-
  - Stem words -> token
  - Remove frequent tokens
  - Remove tokens that do not occur
  - Count occurrences of remaining interesting tokens

Workflow for topic modeling





13.8.8 Text Generation

Text generation is the generation of natural language texts by computer

How?

Ans: Neural networks in NLP

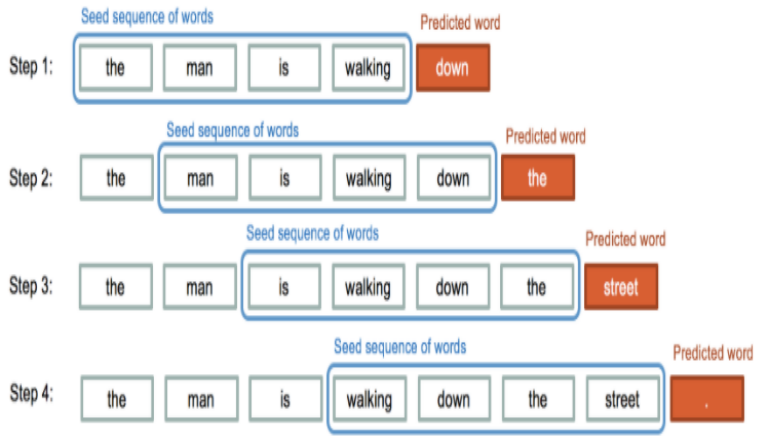
We can pre-process our data and assign every word some ID.

0 1 2 3

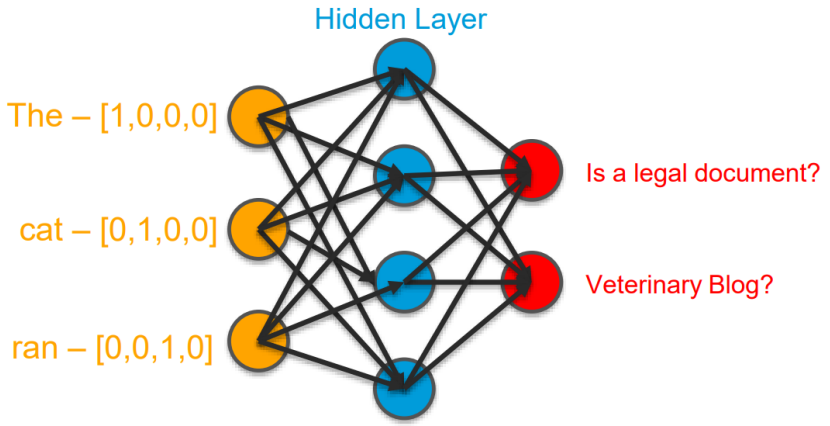
The cat ran fast.

- Then we can use these IDs as inputs to our training by converting them to one-hot-encoded vectors.
- If we want to do classification, we are can then feed it through our neural net.

Using LSTM, RNN, Keras, Markov chain models, text generation models can be implemented in Python and R



A seed sequence



13.9 Examples

- 13.9.1 Preprocessing Texts
- 13.9.2 Document Term Frequency
- 13.9.3 Similarity
- 13.9.4 Document classification
- 13.9.5 Topic Modeling
- 13.9.6 Document Clustering

13.9.1 Example on Preprocessing

Structure the raw text given below using text wrangling methods – tokenization, stemming, lemmatization and cleaning.

- Natural science is a branch of science concerned with the description, prediction, and understanding of natural phenomena, based on empirical evidence from observation and experimentation. Mechanisms such as peer review and repeatability of findings are used to try to ensure the validity of scientific advances.
Natural science can be divided into two main branches: life science and physical science. Life science is alternatively known as biology, and physical science is subdivided into branches: physics, chemistry, astronomy and Earth science.
These branches of natural science may be further divided into more specialized branches (also known as fields). As empirical sciences, natural sciences use tools from the formal sciences, such as mathematics and logic, converting information about nature into measurements which can be explained as clear statements of the "laws of nature".

13.9.1 Example on Preprocessing

For performing text wrangling operations, we have to import the raw text using the ‘tm’ package in R for text mining.

```
install.packages(‘tm’)
library(tm)
text <- (c(“Natural science is a .. of the “laws of nature”.”))
text

Output:
[1] "Natural science is a branch .. of the “laws of nature”.
```

Tokenization

Tokenization can be done in many ways such characters, *words*, *sentences*, *n-gram* etc. based on the delimiter like white space, ‘,’, ‘.’, ‘?’, ‘!’ etc. Tokenization is executed using the ‘tokenizers’ package.

```
Install.packages(‘tokenizers’)
library(tokenizers)
```

13.9.1 Example on Preprocessing

Character Tokenization

```
tokenize_characters(text)
Output:
[1]  "n" "a" "t" "u" "r" "a" "l" "s" "c" "i" "e" "n" "c" "e " "i" "s" "a" "b" "r" "a" "n" "c" "h" "o"
"f"
```

Word tokenization

```
tokenize_words(text)
Output:
[1]  "natural"  "science"  "is"  "a"  "branch"  "of" "science" "concerned"  "with" "the"
```

Sentence tokenization

```
tokenize_sentences(text)

Output:
[2] "Mechanisms such as peer review and repeatability of findings are used to try to ensure the
validity of scientific advances."
[3] "Natural science can be divided into two main branches: life science and physical science."
```

N-gram tokenization

```
tokenize_ngrams(text, n = 2)

Output:
[1] "natural science"  "science is"  "is a"   "a branch"
```

13.9.1 Example on Preprocessing

Stemming and lemmatization

Stemming algorithm cuts off the beginning or end of the word considering the common prefixes and suffixes found in an inflected word. Whereas, lemmatization looks at the morphological analysis of each word to find extract the proper lemma. The 'textstem' package is used for performing stemming and lemmatization.

```
install.packages('textstem')
library(textstem)
stem_strings(text)
Output:
[1]: Natur scienc i a branch of scienc concern with the descript, predict, and understand of natur
phenomena, base on empir    // description & prediction stemmed to descript & predict

Lemmatize_string(text)
Output:
[1]: Natural science be a branch of science concern with the description, prediction, and
understand of natural    // understanding lemmatized to understand
```

Note: Lemmatization and stemming may not work effectively every time. Eg. "Science" has been stemmed to "scienc" which is wrong.

13.9.1 Example on Preprocessing

Cleaning the data

Cleaning the data for text mining depicts removal of *stopwords, numbers, punctuations, whitespaces* etc. to make the text ready for machine understanding. The 'tm' package provides methods to clean the raw data for text mining.

```
require('tm')
text_stopword <- removeWords(text,stopwords())
text_stopword

Output: [1]: Natural science  branch  science concerned  description  // stopwords like 'is', 'a' have been removed from raw text

text_num <- removeNumbers(text_stopword)
text_num
text_pun <- removePunctuation(text_stopword)
text_pun

Output: [1]: known  fields As empirical sciences  // punctuations have been removed
```

13.9.2 Example of Document term Frequency

Build a corpus of the given raw text and clean the data. Calculate the document term frequency of the corpus

- Natural science is a branch of science concerned with the description, prediction, and understanding of natural phenomena, based on empirical evidence from observation and experimentation. Mechanisms such as peer review and repeatability of findings are used to try to ensure the validity of scientific advances.
Natural science can be divided into two main branches: life science and physical science. Life science is alternatively known as biology, and physical science is subdivided into branches: physics, chemistry, astronomy and Earth science.
These branches of natural science may be further divided into more specialized branches (also known as fields). As empirical sciences, natural sciences use tools from the formal sciences, such as mathematics and logic, converting information about nature into measurements which can be explained as clear statements of the "laws of nature".

13.9.2 Example of Document term Frequency

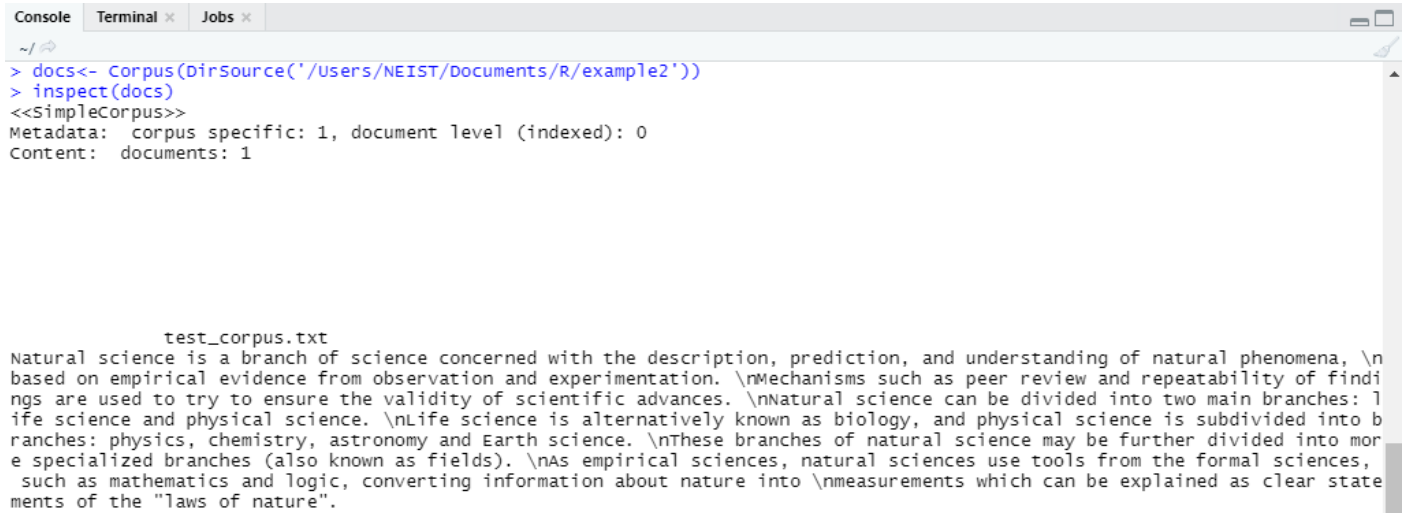
Building a corpus of raw data

For performing text wrangling operations, we have to import the raw text using the 'tm' package in R for text mining. In this example the raw data is saved as "test.txt" in a folder R/example2

Reading the corpus

```
install.packages('tm')
library(tm)
docs<- Corpus(DirSource('/Users/NEIST/Documents/R/example2'))
inspect(docs)        // docs is the corpus of raw data and using 'inspect' the corpus is being read
```

Output



13.9.2 Example of Document term Frequency

Cleaning the corpus

While cleaning the corpus we first convert the text into lower case and then remove punctuation, stop words, whitespace

```
docs_clean <- tm_map(docs, tolower)     // lower case
inspect(docs_clean)
```

Output: natural science is a branch of science concerned with the description, prediction, and understanding of natural phenomena, \nbased on empirical…

```
docs_clean <-tm_map(docs_clean, removePunctuation)      // remove punctuations
inspect(docs_clean)
```

Output: natural science is a branch of science concerned with the description prediction and understanding of natural phenomena \nbased on empirical…

```
docs_clean <- tm_map(docs_clean, removeWords, stopwords())     // remove stopwords
inspect(docs_clean)
```

Output: natural science branch science concerned description prediction understanding natural phenomena \nbased empirical

13.9.2 Example of Document term Frequency

Cleaning the corpus

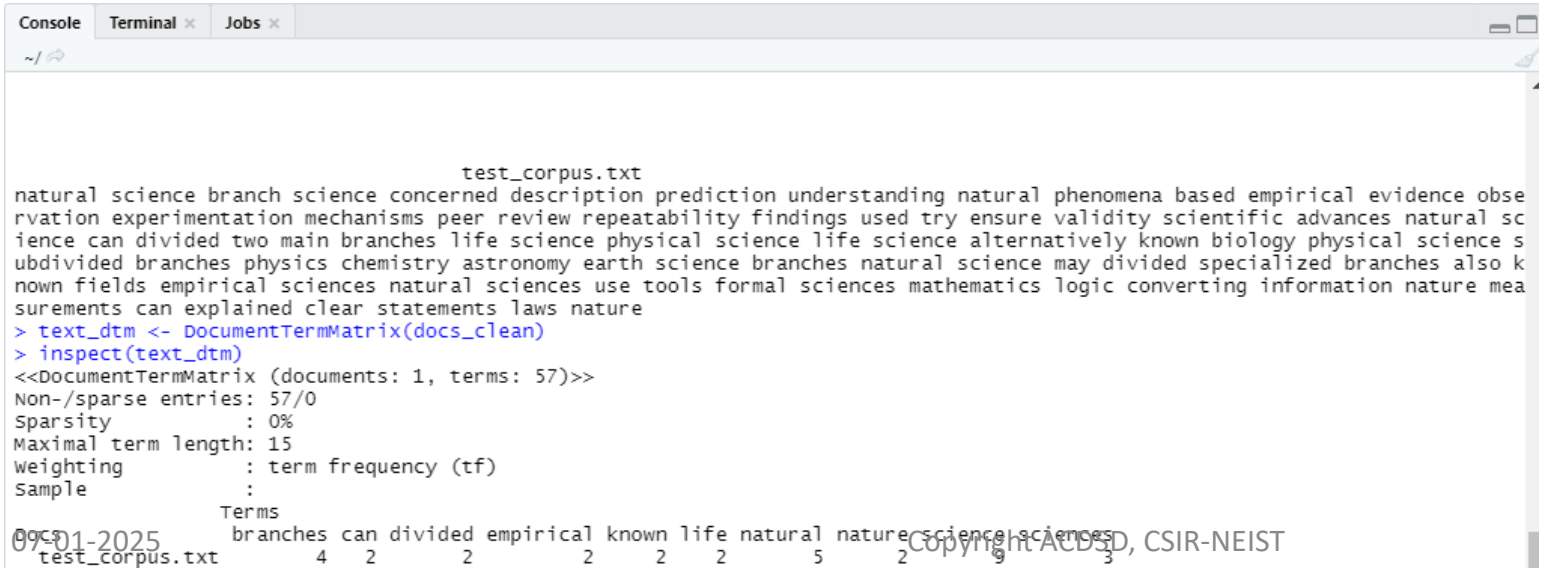
```
docs_clean <- tm_map(docs_clean, stripWhitespace) #remove whitespace
inspect(docs_clean)
```

Output: natural science is a branch of science concerned with the description prediction and understanding of natural phenomena based on empirical …

Document term frequency

```
text_dtm <- DocumentTermMatrix(docs_clean)
inspect(text_dtm)
```

Output:



13.9.3 Example on frequency and similarity

Consider two corpus and compute the similarity between the two documents and find the key words of the two documents

Two corpus have been considered as given below:

corpus_1 : *In the fields of medicine, biotechnology and pharmacology, drug discovery is the process by which new candidate medications are ……. some hope of pharmacotherapeutic advances.*

corpus_2: *The idea that the effect of a drug in the human body is mediated by specific interactions of the drug molecule with biological macromolecules, ……. known as reverse pharmacology and is the most frequently used approach today.*

13.9.3 Example on frequency and similarity

Reading the two corpus

```
docs<- Corpus(DirSource('/Users/NEIST/Documents/R/example3'))
inspect(docs)
```

Output:

corpus_1.txt : In the fields of medicine, biotechnology and pharmacology …

corpus_2.txt : The idea that the effect of a drug in the human body is …

Cleaning the data

```
// create a content transformer to modify the content of an R object
toSpace <- content_transformer(function(x, pattern){return(gsub(pattern, " ",x ))})

docs <- tm_map(docs,toSpace,"-")
docs <- tm_map(docs,toSpace,":")
docs <- tm_map(docs,toSpace,",")
docs <- tm_map(docs,toSpace,"'")
docs <- tm_map(docs,toSpace," _")
docs <- tm_map(docs,toSpace,"\n")
```

13.9.3 Example on frequency and similarity

Cleaning the data

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

Output:

corpus_1.txt: fields medicine biotechnology pharmacology drug discovery process new candidate medications discovered historically drugs discovered ..

corpus_2.txt: idea effect drug human body mediated specific interactions drug molecule biological macromolecules proteins nucleic acids cases led scientists ..

Document term matrix

```
install.packages('SnowballC')
library(SnowballC)
dtm <- DocumentTermMatrix(docs)
inspect(dtm[1:2,1:50])
```

13.9.3 Example on frequency and similarity

Output Document Term Matrix:

```
> library(SnowballC)
> dtm <- DocumentTermMatrix(docs)
> inspect(dtm[1:2,1:100])
<<DocumentTermMatrix (documents: 2, terms: 100)>>
Non-/sparse entries: 122/78
Sparsity           : 39%
Maximal term length: 15
Weighting          : term frequency (tf)
Sample             :
             Terms
Docs          active biological compounds development discovery drug drugs human known large
  corpus_1.txt      1          2         1           3         7    8     3     1     2     4
  corpus_2.txt      3          5         3           1         3    6     1     3     4     1
```

Find Association

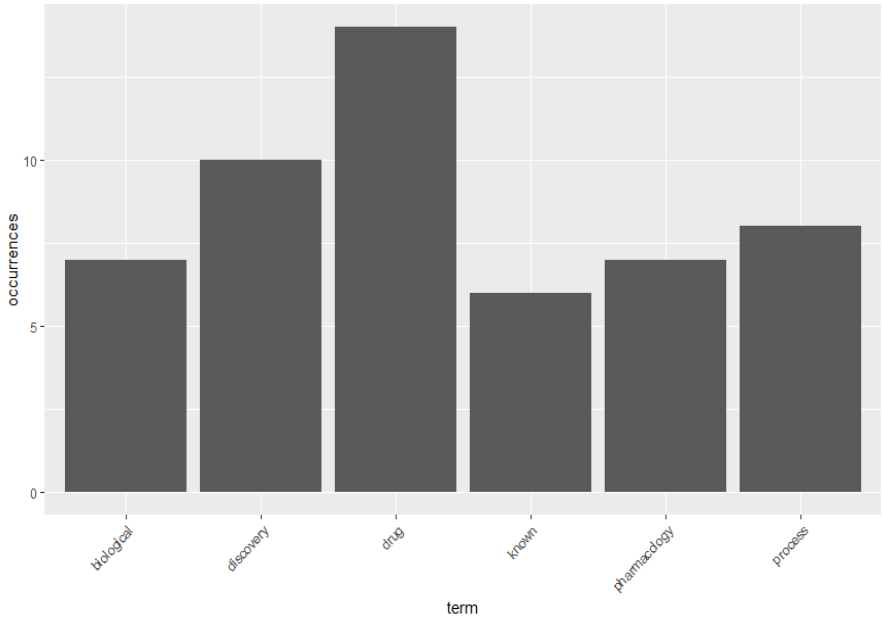
```
findAssocs(dtm,"medicine",0.6)
```

```
> findAssocs(dtm,"medicine",0.6)
$medicine
     academia      advances      affinity       allowed       animals   application
            1             1             1             1             1             1
     approval       balance         basic        become       billion bioavailability
            1             1             1             1             1             1
 biotechnology        called           can     candidate       capital    capitalists
            1             1             1             1             1             1
        cells       century      clinical          come    commercial        common
            1             1             1             1             1             1
 communication     companies       complex      compound      continue  corporations
            1             1             1             1             1             1
```

13.9.3 Example on frequency and similarity

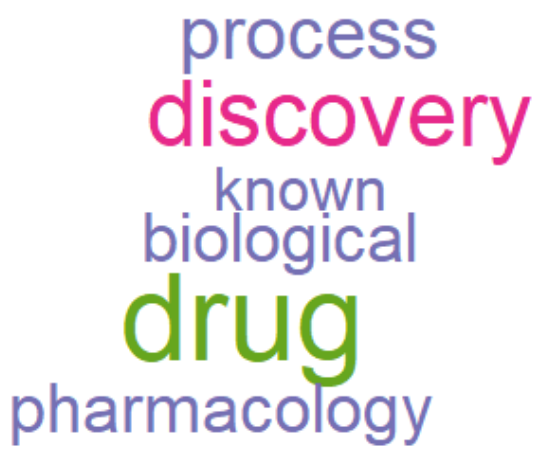
Plot high frequency words

```
install.packages('ggplot2')
library(ggplot2)
wf =data.frame(term=names(freq),occurrences=freq)
p <-ggplot(subset(wf, freq>5), aes(term, occurrences))
p <- p + geom_bar(stat = "identity")
p <- p + theme(axis.text.x = element_text(angle = 45,
hjust = 1))
P
```



Wordcloud of frequency words

```
install.packages('wordcloud')
library(wordcloud)
set.seed(42)
wordcloud(names(freq),freq, min.freq = 6)
wordcloud(names(freq),freq, min.freq = 6, colors =
brewer.pal(5,"Dark2"))
```



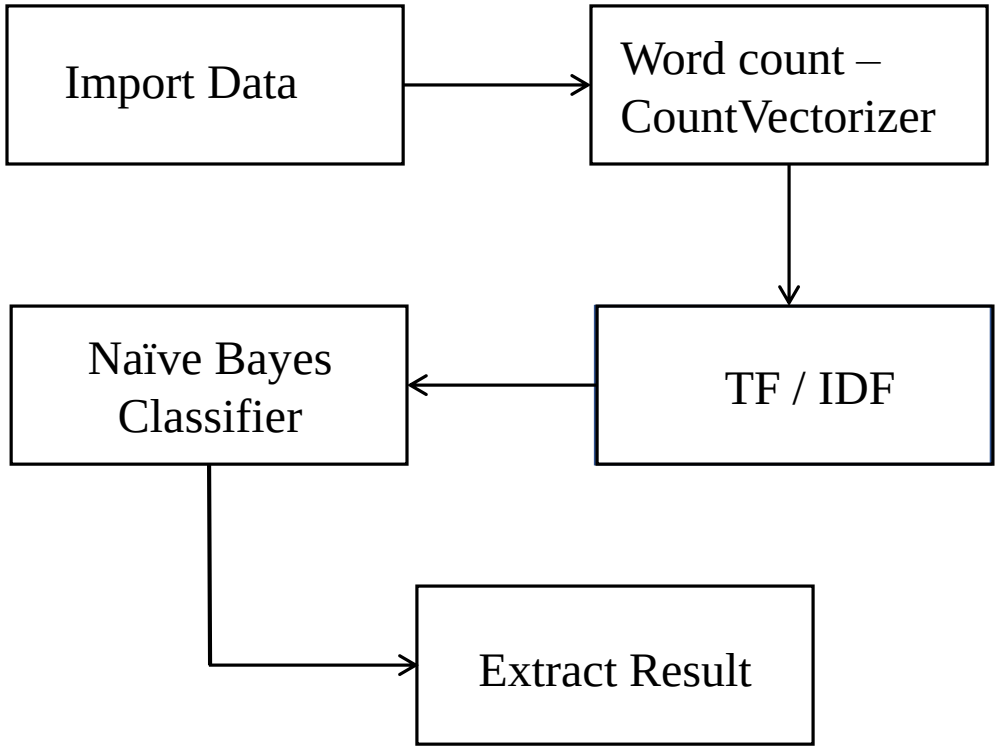
13.9.4 Example on Document Classification

A dataset containing 20 categories with around 11500 observations for training and 7300 observations for testing has been extracted. Design a document classification model and analyse its efficiency. Also, enable the model to predict category of new unknown data by user.

 # Dataset details

Total observations: 18846
Total labelled categories: 20
Categories:
'alt.atheism', 'comp.graphics', 'comp.os.ms-windows.misc', 'comp.sys.ibm.pc.hardware', 'comp.sys.mac.hardware', 'comp.windows.x', 'misc.forsale', 'rec.autos', 'rec.motorcycles', 'rec.sport.baseball', 'rec.sport.hockey', 'sci.crypt', 'sci.electronics', 'sci.med', 'sci.space', 'soc.religion.christian', 'talk.politics.guns', 'talk.politics.mideast', 'talk.politics.misc', 'talk.religion.misc'

Workflow



13.9.4 Example on Document Classification


```
# Creating libraries
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()

# Importing data
from sklearn.datasets import fetch_20newsgroups
data = fetch_20newsgroups()
data.target_names


categories = ['alt.atheism', 'comp.graphics', 'comp.os.ms-
windows.misc', 'comp.sys.ibm.pc.hardware',
'comp.sys.mac.hardware', 'comp.windows.x',  'misc.forsale',
'rec.autos', 'rec.motorcycles', 'rec.sport.baseball',
'rec.sport.hockey', 'sci.crypt', 'sci.electronics', 'sci.med',
'sci.space', 'soc.religion.christian', 'talk.politics.guns',
'talk.politics.mideast', 'talk.politics.misc', 'talk.religion.misc']
train = fetch_20newsgroups(subset='train', categories=
categories)
test = fetch_20newsgroups(subset= 'test', categories=
categories)
```

```
Out[7]: ['alt.atheism',
         'comp.graphics',
         'comp.os.ms-windows.misc',
         'comp.sys.ibm.pc.hardware',
         'comp.sys.mac.hardware',
         'comp.windows.x',
         'misc.forsale',
         'rec.autos',
         'rec.motorcycles',
         'rec.sport.baseball',
         'rec.sport.hockey',
         'sci.crypt',
         'sci.electronics',
         'sci.med',
         'sci.space',
         'soc.religion.christian',
         'talk.politics.guns',
         'talk.politics.mideast',
         'talk.politics.misc',
         'talk.religion.misc']
```

Fetching the train and test data set in separate variables

13.9.4 Example on Document Classification


```
# Importing packages
from sklearn.feature_extraction.text import
TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline

# Creating Naïve Bayes Multinomial model
model = make_pipeline(TfidfVectorizer(),
MultinomialNB())

# Training the model with train data
model.fit(train.data, train.target)

# Creating labels for test data
labels = model.predict(test.data)
```

Multinomial Naïve Bayes model

Count Vectorizer
- count words

TFIDF Transformer
- frequency

TFIDF vectorizer
- Count words
- frequency

TFIDF Vectorizer

MultiNomial NB

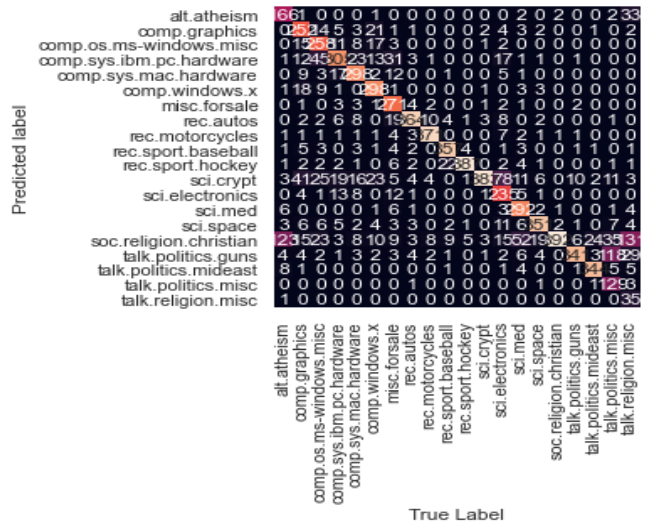
Make pipeline with a object

Object model = 'TFIDF vectorizer' 'MultiNomial NB()'

13.9.4 Example on Document Classification


```
#Import packages
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix

#Accuracy computation
print('Accuracy achieved is ' + str(np.mean(labels = = test.target)))
print(metrics.classification_report(test.target, labels, target_names=test.target_names))

# Confusion matrix and heap map
mat = confusion_matrix(test.target, labels)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False
        , xticklabels=train.target_names
        , yticklabels=train.target_names)

# Plotting heatmap
plt.xlabel('True Label')
plt.ylabel('Predicted label')
```

```
Accuracy achieved is 0.7738980350504514
                          precision    recall  f1-score   support

             alt.atheism       0.80      0.52      0.63       319
           comp.graphics       0.81      0.65      0.72       389
 comp.os.ms-windows.misc       0.82      0.65      0.73       394
comp.sys.ibm.pc.hardware       0.67      0.78      0.72       392
   comp.sys.mac.hardware       0.86      0.77      0.81       385
          comp.windows.x       0.89      0.75      0.82       395
            misc.forsale       0.93      0.69      0.80       390
               rec.autos       0.85      0.92      0.88       396
         rec.motorcycles       0.94      0.93      0.93       398
      rec.sport.baseball       0.92      0.90      0.91       397
        rec.sport.hockey       0.89      0.97      0.93       399
               sci.crypt       0.59      0.97      0.74       396
         sci.electronics       0.84      0.60      0.70       393
                 sci.med       0.92      0.74      0.82       396
               sci.space       0.84      0.89      0.87       394
  soc.religion.christian       0.44      0.98      0.61       398
      talk.politics.guns       0.64      0.94      0.76       364
   talk.politics.mideast       0.93      0.91      0.92       376
      talk.politics.misc       0.96      0.42      0.58       310
      talk.religion.misc       0.97      0.14      0.24       251

                accuracy                           0.77      7532
               macro avg       0.83      0.76      0.76      7532
            weighted avg       0.82      0.77      0.77      7532
```



13.9.4 Example on Document Classification

 # Predicting new data based on trained model

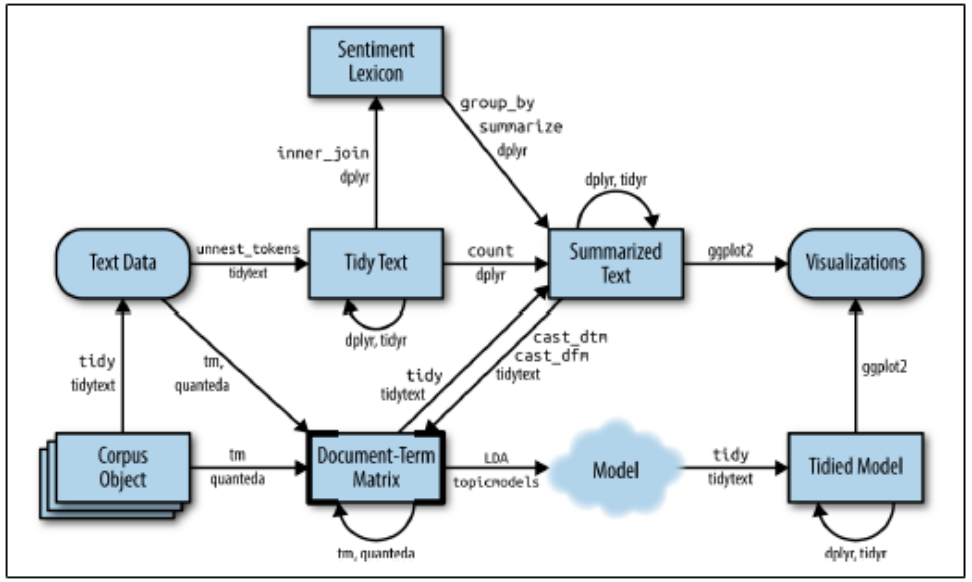
```
def predict_category(s, train=train, model=model):
 pred = model.predict([s])
 return train.target_names[pred[0]]


predict_category('Honda')
predict_category('BMW and Audi')
predict_category('President of USA')
predict_category('RAM is primary')
predict_category('National game is Hockey')
```

```
In [22]: predict_category('Honda')
Out[22]: 'rec.motorcycles'

In [25]: predict_category('BMW and Audi')
Out[25]: 'rec.autos'

In [31]: predict_category('President of USA')
Out[31]: 'talk.politics.misc'

In [33]: predict_category('RAM is primary')
Out[33]: 'comp.sys.ibm.pc.hardware'

In [34]: predict_category('National game is Hockey')
Out[34]: 'rec.sport.hockey'
```

13.9.5 Example on Topic Modeling

Using the "Associated Press dataset" contiaing 2246 news articles from the First Text Retrieval Conference, design a LDA based model for topic modeling in R.

```
# Packages and libraries

install.packages("tidytext")
install.packages("reshape2")
install.packages("topicmodels")

library(tm)
library(dplyr)
library(reshape2)
library(tidytext)
library(topicmodels)
library(ggplot2)

# Loading dataset

data("AssociatedPress", package = "topicmodels")
AssociatedPress
```



Workflow for topic modeling

```
> AssociatedPress
<<DocumentTermMatrix (documents: 2246, terms: 10473)>>
Non-/sparse entries: 302031/23220327
Sparsity           : 99%
Maximal term length: 18
Weighting          : term frequency (tf)
```

13.9.5 Example on Topic Modeling

 # Extract terms in the dataset

```
terms <- Terms(AssociatedPress)
head(terms)
tail(terms)
```

```
> terms <- Terms(AssociatedPress)
> head(terms)
[1] "aaron"      "abandon"    "abandoned"  "abandoning" "abbott"     "abboud"
> tail(terms)
[1] "zinoviev" "zone"     "zones"    "zoo"      "zubal"    "zurich"
```

```
ap_td <- tidy(AssociatedPress)
ap_td
```

Tyding the data for one document per row

```
# A tibble: 302,031 x 3
   document term        count
      <int> <chr>       <dbl>
 1        1 adding          1
 2        1 adult           2
 3        1 ago             1
 4        1 alcohol         1
 5        1 allegedly       1
 6        1 allen           1
 7        1 apparently      2
 8        1 appeared        1
 9        1 arrested        1
10        1 assault         1
# ... with 302,021 more rows
```

 # Creating the LDA model

```
ap_lda <- LDA(AssociatedPress, k = 3, control =
list(seed = 1234))
ap_lda
```

A LDA_VEM (Variation expectation-maximization) topic model with 3 topics.

 # Per-topic-per-word probabilities (*β*)

```
ap_topics <- tidy(ap_lda, matrix = "beta")
ap_topics
```

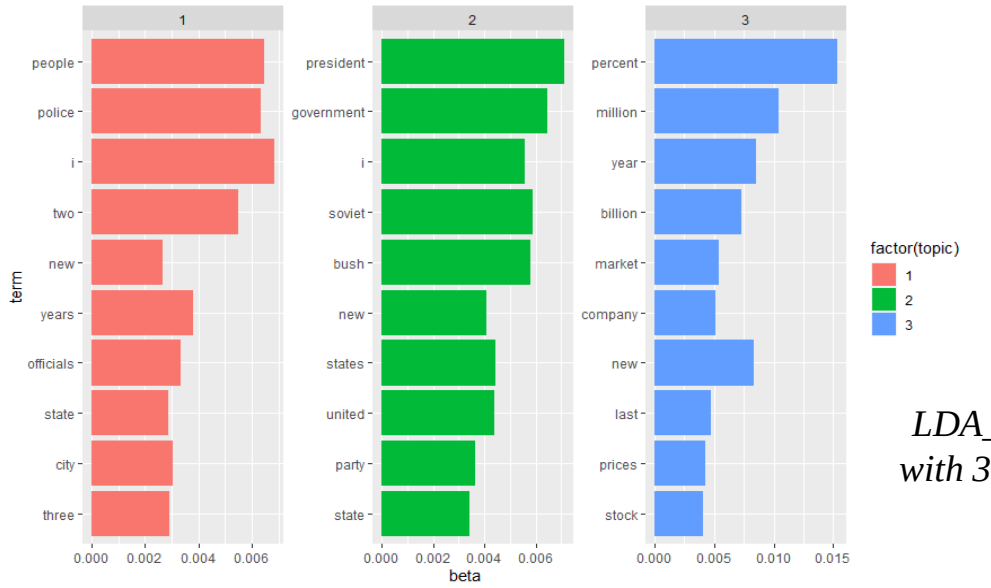
```
# A tibble: 31,419 x 3
   topic term                beta
   <int> <chr>              <dbl>
 1     1 aaron       0.0000248
 2     2 aaron       0.0000192
 3     3 aaron       0.0000257
 4     1 abandon     0.000000161
 5     2 abandon     0.0000609
 6     3 abandon     0.0000434
 7     1 abandoned   0.000203
 8     2 abandoned   0.0000323
 9     3 abandoned   0.0000180
10     1 abandoning  0.000000000843
# ... with 31,409 more rows
```

13.9.5 Example on Topic Modeling

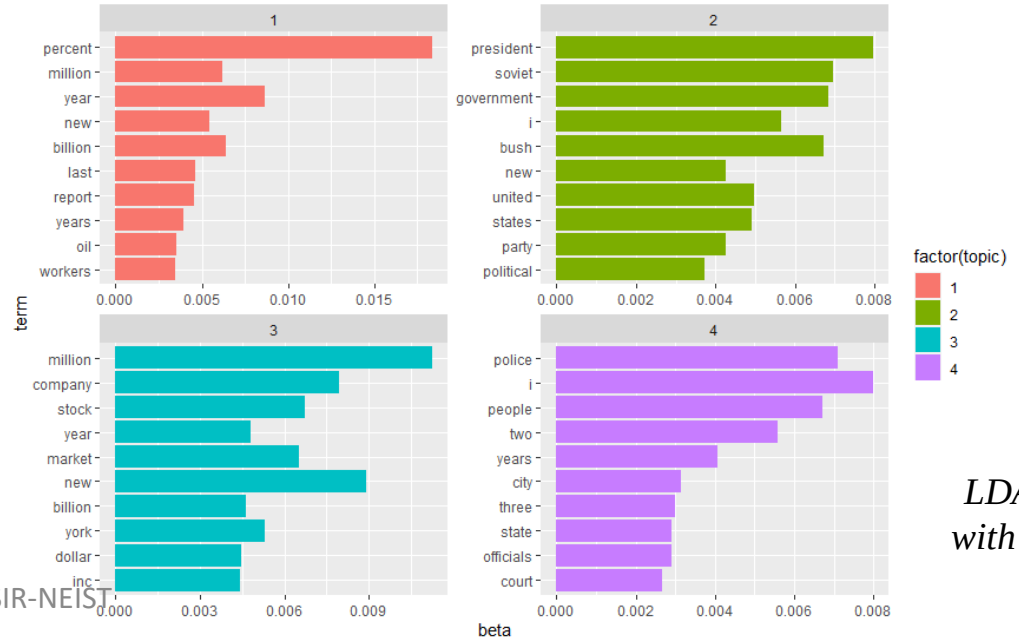

```
# Identifying top terms per topic
ap_top_terms <- ap_topics %>%
  group_by(topic) %>%
  top_n(10, beta) %>%
  ungroup() %>%
  arrange(topic, -beta)
ap_top_terms

# Visualization
ap_top_terms %>%
  mutate(term = reorder(term, beta)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  coord_flip()
```

```
# A tibble: 30 x 3
   topic term          beta
   <int> <chr>        <dbl>
 1     1 i          0.00684
 2     1 people     0.00645
 3     1 police     0.00632
 4     1 two        0.00550
 5     1 years      0.00381
 6     1 officials  0.00335
 7     1 city       0.00302
 8     1 three      0.00291
 9     1 state      0.00286
10     1 new        0.00264
# ... with 20 more rows
```

Top 10 terms in Topic 1



LDA_VEM with 3 topics



LDA_VEM with 4 topics

13.9.6 Example on Document Clustering

The "*nih_sample*" dataset has 100 observations with 44 variables on the grant abstracts, create a document clustering model in using Hierarchical and K-Means clustering in R to cluster these observations as per content.

```
# Packages and Libraries
install.packages("textmineR")
library(textmineR)
library(cluster)

# Load nih_sample dataset
data(nih_sample)
nih_sample
head(nih_sample)

# Preprocessing and DTM calculation
dtm <- CreateDtm(doc_vec = nih_sample$ABSTRACT_TEXT,
                 doc_names = nih_sample$APPLICATION_ID,
                 ngram_window = c(1, 2),
                 stopword_vec = c(stopwords::stopwords("en"),
                                  stopwords::stopwords(source = "smart")),
                 lower = TRUE,
                 remove_punctuation = TRUE,
                 remove_numbers = TRUE,
                 verbose = FALSE,
                 cpus = 2)
```

```
#> 'data.frame':    100 obs. of  44 variables:
#>  $ APPLICATION_ID        : chr  "8693991" "8693362" "8607498" "8697008" ...
#>  $ ABSTRACT_TEXT         : chr  "Methamphetamine (MA) is remarkably addictive and relapse to excessive
#>  $ ACTIVITY              : chr  "P50" "R01" "R21" "K01" ...
#>  $ ADMINISTERING_IC      : chr  "DA" "GM" "AI" "AI" ...
#>  $ APPLICATION_TYPE      : chr  "5" "2" "5" "5" ...
#>  $ ARRA_FUNDED           : chr  "N" "N" "N" "N" ...
#>  $ AWARD_NOTICE_DATE     : chr  "06/30/2014" "05/27/2014" "01/03/2014" "07/11/2014" ...
#>  $ BUDGET_START          : chr  "07/01/2014" "06/01/2014" "02/01/2014" "08/01/2014" ...
#>  $ BUDGET_END            : chr  "06/30/2015" "04/30/2015" "01/31/2015" "07/31/2015" ...
#>  $ CFDA_CODE             : chr  "" "859" "855" "855" ...
#>  $ CORE_PROJECT_NUM      : chr  "P50DA018165" "R01GM085047" "R21AI100696" "K01AI100681" ...
#>  $ ED_INST_TYPE          : chr  "" "" "SCHOOLS OF MEDICINE" "SCHOOLS OF MEDICINE" ...
#>  $ FOA_NUMBER            : chr  "PAR-10-189" "PA-11-260" "PA-11-261" "PA-11-190" ...
#>  $ FULL_PROJECT_NUM      : chr  "5P50DA018165-08" "2R01GM085047-05" "5R21AI100696-02" "5K01AI100681-03"
#>  $ FUNDING_ICs           : chr  "NIDA:212488\\" "NIGMS:326324\\" "NIAID:209613\\" "NIAID:132182\\" ...
#>  $ FUNDING_MECHANISM     : chr  "Research Centers" "Research Projects" "Research Projects" "Other Rese
#>  $ FY                    : chr  "2014" "2014" "2014" "2014" ...
#>  $ IC_NAME               : chr  "NATIONAL INSTITUTE ON DRUG ABUSE" "NATIONAL INSTITUTE OF GENERAL MEDI
#>  $ NIH_SPENDING_CATS     : chr  "" "" "" "" ...
#>  $ ORG_CITY              : chr  "PORTLAND" "SEATTLE" "NEW HAVEN" "BALTIMORE" ...
#>  $ ORG_COUNTRY           : chr  "UNITED STATES" "UNITED STATES" "UNITED STATES" "UNITED STATES" ...
#>  $ ORG_DEPT              : chr  "" "" "MICROBIOLOGY/IMMUN/VIROLOGY" "EMERGENCY MEDICINE" ...
#>  $ ORG_DISTRICT          : chr  "03" "" "03" "07" ...
#>  $ ORG_DUNS              : chr  "" "" "" "" ...
#>  $ ORG_FIPS              : chr  "" "US" "US" "US" ...
```

Preprocessing of the text like stop word removal, remove punctuations, numbers along with DTM is computed

13.9.6 Example on Document Clustering


```
# construct the matrix of term counts to get the IDF vector
tf_mat <- TermDocFreq(dtm)
tf_mat

# TF-IDF and cosine similarity
tfidf <- t(dtm[ , tf_mat$term ]) * tf_mat$idf
tfidf <- t(tfidf)
tfidf
csim <- tfidf / sqrt(rowSums(tfidf * tfidf))
csim <- csim %*% t(csim)
csim
cdist <- as.dist(1 - csim)
cdist
```

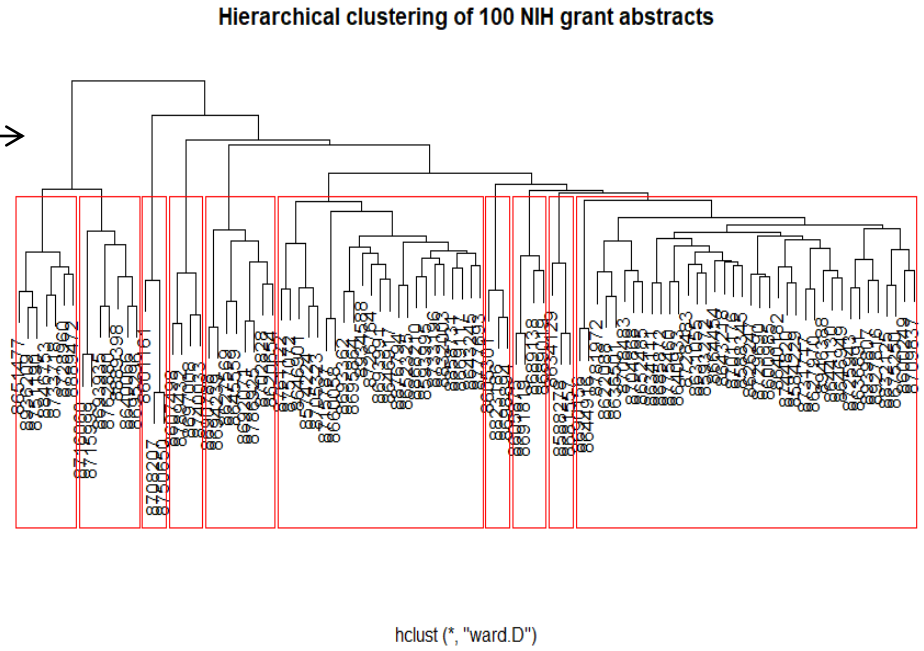
```
> tf_mat
                                                 term term_freq doc_freq      idf
aaas                                             aaas         1        1 4.60517
aaas_review                               aaas_review         1        1 4.60517
abating                                     abating         1        1 4.60517
abating_imperative                 abating_imperative         1        1 4.60517
abilities_director                 abilities_director         1        1 4.60517
abilities_variety                   abilities_variety         1        1 4.60517
abilities_warrants                 abilities_warrants         1        1 4.60517
ability_drive                           ability_drive         1        1 4.60517
ability_effectively               ability_effectively         1        1 4.60517
ability_enhance                       ability_enhance         1        1 4.60517
ability_evade                           ability_evade         1        1 4.60517
ability_evaluate                     ability_evaluate         1        1 4.60517
ability_extrapolate               ability_extrapolate         1        1 4.60517
ability_form                             ability_form         1        1 4.60517
ability_induce                         ability_induce         1        1 4.60517
ability_interface                   ability_interface         1        1 4.60517
ability_invade                         ability_invade         1        1 4.60517
ability_lung                             ability_lung         1        1 4.60517
ability_mast                             ability_mast         1        1 4.60517
ability_mechanism                   ability_mechanism         1        1 4.60517
ability_modulate                     ability_modulate         1        1 4.60517
ability_ms                                 ability_ms         1        1 4.60517
ability_neuroprotect         ability_neuroprotect         1        1 4.60517
ability_promote                       ability_promote         1        1 4.60517
ability_provide                       ability_provide         1        1 4.60517
ability_tetanic                       ability_tetanic         1        1 4.60517
ability_uppress                       ability_uppress         1        1 4.60517
```

```
> tfidf
100 x 23915 sparse Matrix of class "dgCMatrix"
   [[ suppressing 72 column names 'aaas', 'aaas_review', 'abating' ... ]]
   [[ suppressing 72 column names 'aaas', 'aaas_review', 'abating' ... ]]


8693991 .      .      . . . . . .      . .      . . . . . . . . . . . .      . . . . . .      .      .      . .
 . . . .
8693362 .      .      . . . . . .      . .      . . . . . . . . . . . .      . . . . . .      .      .      . .
 . . . .
8607498 .      .      . . . . . .      . 4.60517 . . . . . . . . . . . 4.60517 . . . . . .      .      .      . .
 . . . .
8697008 .      .      . . . . . .      . .      . . . . . . . . . . . .      . . . . . . 4.60517 4.60517 4.60517 . .
 . . . .
8725943 .      .      . . . . . .      . .      . . . . . . . . . . . .      . . . . . .      .      .      . .
 . . . .
8574224 .      .      . . . . . .      . .      . . . . . . . . . . . .      . . . . . .      .      .      . .
 . . . .
8716060 4.60517 4.60517 . . . . . 4.60517 . .      . . . . . . . . . . . .      . . . . . .      .      .      . .
```

```
> csim
100 x 100 sparse Matrix of class "dgCMatrix"
   [[ suppressing 46 column names '8693991', '8693362', '8607498' ... ]]
   [[ suppressing 46 column names '8693991', '8693362', '8607498' ... ]]

8693991 1.000000000 0.006513148 0.010720303 0.010303703 0.004655217 0.005642982 0.011498976 0.011741836 0.022051220
8693362 0.006513148 1.000000000 0.003817193 0.021573889 0.005486238 0.024566607 0.020323306 0.020410111 0.007077840
8607498 0.010720303 0.003817193 1.000000000 0.088263763 0.005621670 0.003061398 0.002902403 0.012466140 0.003015135
8697008 0.010303703 0.021573889 0.088263763 1.000000000 0.007901016 0.004921241 0.026466001 0.003114663 0.005081968
8725943 0.004655217 0.005486238 0.005621670 0.007901016 1.000000000 0.002273049 0.001018577 0.010048627 0.015762032
8574224 0.005642982 0.024566607 0.003061398 0.004921241 0.002273049 1.000000000 0.004971753 0.001579781 0.009564653
8716060 0.011498976 0.020323306 0.002902403 0.026466001 0.001018577 0.004971753 1.000000000 0.001169767 0.006469793
8692071 0.011741836 0.020410111 0.012466140 0.003114663 0.010048627 0.001579781 0.001169767 1.000000000 0.015430710
8644130 0.022051220 0.007077840 0.003015135 0.005081968 0.015762032 0.009564653 0.006469793 0.015430710 1.000000000
8702828 0.012139412 0.020635981 0.004901377 0.005377208 0.010838001 0.002274421 0.006492198 0.017409940 0.010001404
8823186 0.012568221 0.013050556 0.004096145 0.009313754 0.008245862 0.018240429 0.005827307 0.005658745 0.008117314

8693991 0.012139412 0.012568221 0.009681012 0.0074841694 0.0132990925 0.008590055 0.012190813 0.014674426
8693362 0.020635981 0.013050556 0.002540121 0.0088908516 0.0066414960 0.005270855 0.009913993 0.007937815
8607498 0.004901377 0.004096145 0.005458671 0.0024686157 0.0009036538 0.008408414 0.003021966 0.006406426
8697008 0.005377208 0.009313754 0.001922033 0.0131909848 0.0080781748 0.004634600 0.006822480 0.012267914
8725943 0.010838001 0.008245862 0.011797439 0.0011908387 0.0065731323 0.018838324 0.011709161 0.014450496
8574224 0.002274421 0.018240429 0.007840920 0.0145824671 0.0200926127 0.008098362 0.016116610 0.003291868
8716060 0.006492198 0.005827307 0.002143165 0.0055683496 0.0054639711 0.003234241 0.004740378 0.002677571
8692071 0.017409940 0.005658745 0.006996507 0.0028543173 0.0057353203 0.006295405 0.010782507 0.016647340
8644130 0.010001404 0.008117314 0.026705607 0.0033857928 0.0141995276 0.020637621 0.008467785 0.008994570
8702828 1.000000000 0.005148903 0.006742819 0.0003369757 0.0105566646 0.004352799 0.006676108 0.008903608
8823186 0.005148903 1.000000000 0.006881872 0.0193879454 0.0097038321 0.024846257 0.098631221 0.003722667
```

13.9.6 Example on Document Clustering

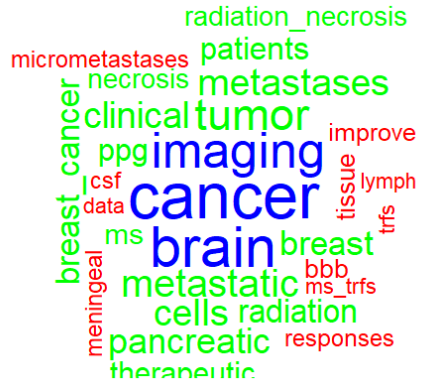

```
# Heirarchcial clustering
hc <- hclust(cdist, "ward.D")
clustering <- cutree(hc, 10)
plot(hc, main = "Hierarchical clustering of 100 NIH grant abstracts",
     ylab = "", xlab = "", yaxt = "n")
rect.hclust(hc, 10, border = "red")

# Drop all words that don't appear in the cluster
p_words <- colSums(dtm) / sum(dtm)
cluster_words <- lapply(unique(clustering), function(x){
  rows <- dtm[ clustering == x , ]
  rows <- rows[ , colSums(rows) > 0 ]
  colSums(rows) / sum(rows) - p_words[ colnames(rows) ]
 })

# Create a summary table of the top 5 words defining each cluster
cluster_summary <- data.frame(cluster = unique(clustering),
                      size = as.numeric(table(clustering)),
                      top_words = sapply(cluster_words, function(d){
                        paste(
                          names(d)[ order(d, decreasing = TRUE) ][ 1:5 ],
                          collapse = ", ")
                      }),
                      stringsAsFactors = FALSE)
cluster_summary
```




```
> cluster_summary
   cluster size                                        top_words
1        1   23  risk, health, diabetes, intervention, treatment
2        2    4           hiv, inflammation, env, testing, study
3        3   38      cell, infection, determine, cells, function
4        4    7 research, program, cancer, disparities, students
5        5    8        cancer, brain, imaging, tumor, metastatic
6        6    3  microbiome, crc, gut, psoriasis, gut_microbiome
7        7    3                  cdk, nmdar, nmdars, calpain, nrs
8        8    7   research, core, center, support, translational
9        9    3        lung, ipf, expression, cells, methylation
10      10    4        mitochondrial, metabolic, redox, ros, bde
```

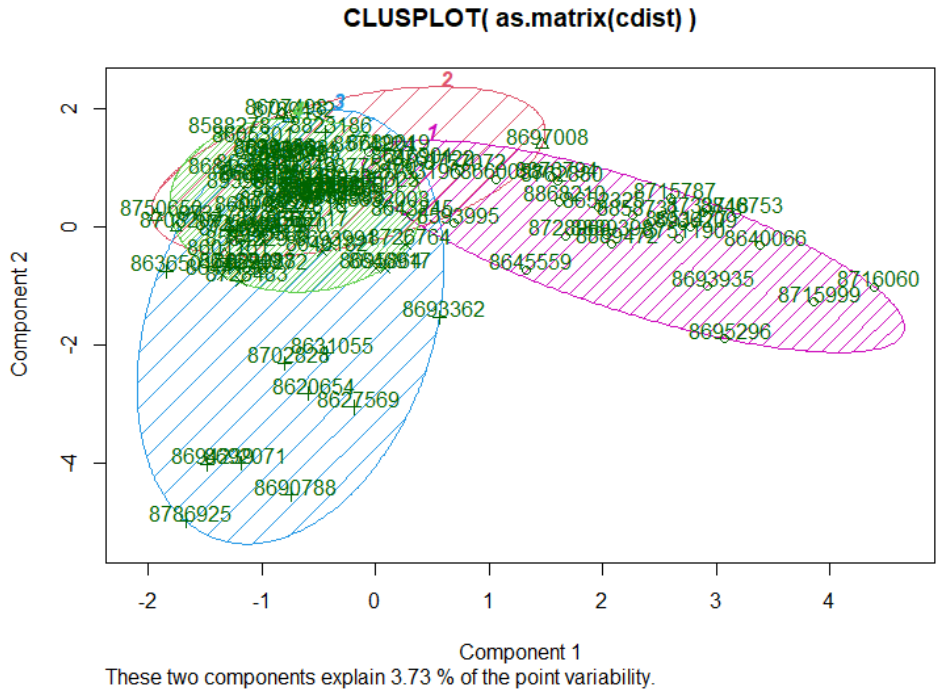
13.9.6 Example on Document Clustering


```
# plot a word cloud of one cluster as an example
wordcloud::wordcloud(words = names(cluster_words[[ 5 ]]),
                     freq = cluster_words[[ 5 ]],
                     max.words = 30,
                     random.order = FALSE,
                     colors = c("red", "green", "blue"),
                     main = "Top words in cluster 100")
```




```
# K-Means clustering (k = 4)
# K-means clustering
set.seed(123)
kc<-kmeans(cdist, 4)
kc
clusplot(as.matrix(cdist), kc$cluster, color=T, shade=T, labels=2, lines=0)
```

```
> kc
K-means clustering with 4 clusters of sizes 27, 7, 20, 46

Cluster means:
    8693991   8693362   8607498   8697008   8725943   8574224   8716060   8692071   8644130   8702828
1 0.9886369 0.9852694 0.9967055 0.9793135 0.9963561 0.9500064 0.9332291 0.9946553 0.9941322 0.9929777
2 0.9886689 0.9903852 0.8151165 0.8231361 0.9892431 0.9945178 0.9921679 0.9911226 0.9908757 0.9939780
3 0.9859042 0.9348479 0.9897499 0.9903084 0.9894955 0.9911038 0.9940559 0.9276704 0.9866661 0.9319135
4 0.9687446 0.9915827 0.9938190 0.9922174 0.9688768 0.9934611 0.9961550 0.9913546 0.9672153 0.9927702
    8823186   8946388   8761122   8690156   8731717   8698894   8622088   8643245   8588278   8832125
1 0.9887897 0.9951041 0.9509248 0.9917582 0.9961535 0.9932549 0.9946170 0.9907747 0.9973674 0.9938578
2 0.9911716 0.9888819 0.9952134 0.9943728 0.9895176 0.9934371 0.9913280 0.9941201 0.9742204 0.9969757
3 0.9327599 0.9908131 0.9930736 0.9908646 0.9866689 0.9313473 0.9865152 0.9944439 0.9947742 0.9950452
4 0.9924665 0.9689768 0.9956627 0.9647375 0.9683526 0.9930906 0.9690270 0.9711064 0.9671011 0.9686541
    8868210   8936209   8781972   8636501   8846753   8775460   8704868   8728483   8627569   8593995
1 0.9413885 0.9407367 0.9943262 0.9932334 0.9381067 0.9906641 0.9962617 0.9967510 0.9848870 0.9506621
2 0.9902827 0.9932896 0.9937258 0.9776034 0.9913135 0.9947065 0.9819737 0.9917633 0.9866678 0.9945818
3 0.9904706 0.9947424 0.9910239 0.9307159 0.9930965 0.9912866 0.9895139 0.9882675 0.9294207 0.9919631
4 0.9915218 0.9955381 0.9677241 0.9875880 0.9947823 0.9706056 0.9724535 0.9709129 0.9887404 0.9937924
    8646901   8858731   8640066   8876794   8744820   8705323   8600249   8636891   8740033   8715787
1 0.9477679 0.9460046 0.9374291 0.9479984 0.9954799 0.9456526 0.9946002 0.9899815 0.9916468 0.9425591
2 0.9919161 0.9944409 0.9924714 0.9947630 0.9902874 0.9914951 0.9952146 0.9891397 0.8182363 0.9975757
3 0.9903975 0.9967818 0.9937750 0.9945020 0.9335101 0.9869551 0.9935633 0.9890919 0.9856815 0.9973642
4 0.9923364 0.9965379 0.9956654 0.9960713 0.9911796 0.9912786 0.9695474 0.9631112 0.9898635 0.9972366
    8708147   8652485   8646917   8786925   8644916   8605875   8789432   8733196   8652325   8729690
1 0.9966345 0.9912809 0.9903485 0.9909213 0.9941222 0.9938994 0.9955169 0.9895677 0.9447680 0.9953869
```



13.10 Exercises

- Given an article based on text mining techniques try to find out the following –
  - Is there any particular word occurring in that article and if yes what is the frequency?
  - On what topic essentially the article is focusing on?

- Given a list of articles do the following using count vectorizer –
  - Cluster all the articles based on a criteria?
  - Classify the articles into corresponding domain?
- Given a list of documents do the following using ML techniques –
  - Clusters the documents as per the topic of the document
  - Explain how the result is different from the classical clustering methods

13.10 Exercises

- Given a list of articles do the following using ML techniques –
  - Cluster all the articles based on a criteria?
  - Classify the articles into corresponding domain?

- Given the following Confusion Matrix.
  - Calculate all the evaluation matrix.
  - Which matrix is correct to use.

	A	**C**	**D**	**S**
A	5	15	0	0
D	0	50	5	0
S	0	0	10	25
C	0	20	0	30

13.11 References

The Topics Covered in This Section

- Manning, C.D., and Schutze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press.

➢ *The book contains all the theory and algorithms needed for building NLP tools. It provides broad but rigorous coverage of mathematical and linguistic foundations, as well as detailed discussion of statistical methods. The book covers collocation finding, word sense disambiguation, probabilistic parsing, information retrieval, and other applications.*

- Bishop, C.M. (2006). *Pattern Recognition and Machine Learning*. Springer.

➢ This book reflects recent developments while providing a comprehensive introduction to the fields of pattern recognition and machine learning. It presents approximate inference algorithms that permit fast approximate answers in situations where exact answers are not feasible.

- Srivastava, A., and Sahami, M. (2009). *Text Mining: Classification, Clustering, and Applications.* Chapman & Hall.

➢ *This book focuses on statistical methods for text mining and analysis. It examines methods to automatically cluster and classify text documents and applies these methods in a variety of areas, including adaptive information filtering, information distillation, and text search.*

- Ingersoll, G.S., Morton, T.S., and Farris, A.L. (2013). *Taming Text: How to Find, Organize, and Manipulate It.* Manning.

➢ *This book is a hands-on, example-driven guide to working with unstructured text in the context of real-world applications. This book explores how to automatically organize text using approaches such as full-text search, proper name recognition, clustering, tagging, information extraction, and summarization.*

- Weiss, S.M., Indurkhya, N., and Zhang, T. (2015). *Fundamentals of Predictive Text Mining*. Springer-Verlag London.

➢ *This book Integrates topics spanning the varied disciplines of data mining, machine learning, databases, and computational linguistics and also provides practical advice for text mining.*

- Raj, S. (2019). *Building Chatbots with Python: Using Natural Language Processing and Machine Learning.* Apress.

➢ *This book presents an introduction to chatbots with vital information on their architecture. It also provides information on natural language processing with the natural language toolkit (NLTK) for building a custom language processing platform for chatbot.*

- Sarkar, D. (2019). *Text Analytics with Python: A Practitioner's Guide to Natural Language Processing*. Apress.

➢ *This book covers the latest state-of-the-art frameworks in NLP, coupled with machine learning and deep learning models for supervised sentiment analysis powered by Python to solve actual case studies.*

- Kwartler, T. (2017). *Text Mining in Practice With R*. Wiley.

➢ *All of the tools needed to perform text mining and how to use them for practical business applications with creative text mining efforts are covered in this book.*

- Silge, J., and Robinson, D. (2017). *Text Mining With R.* O'Reilly.

➢ *This book explores machine learning on text covering clustering, classification and prediction. Text analysis techniques like word embedding, complex tokenization have been explained.*

13.11.2 Video Lectures

- Natural Language Processing | Dan Jurafsky, Christopher Manning

https://www.youtube.com/watch?v=3Dt_yh1mf_U&list=PLQiyVNMpDLKnZYBTUOlSI9mi9wAErFtFm

- CS224N: Natural Language Processing with Deep Learning | Christopher Manning Winter 2019

https://www.youtube.com/watch?v=8rXD5hemo&list=PLoROMvodv4rOhcuXMZkNm7j3fVwBBY42z

- Natural Language Processing | University of Michigan | Dragomir Radeev

https://www.youtube.com/watch?v=n25JjoixM3I&list=PLLssT5z_DsK8BdawOVCCaTCO99Ya58ryR

- Text Mining and Analytics [FULL COURSE] | UIUC | ChengXiang Zhai

https://www.youtube.com/watch?v=Uqs0GewlMkQ&list=PLLssT5z_DsK8Xwnh_0bjN4KNT81bekvtt

- Natural Language Processing With Python and NLTK | Harrison Kinsley

https://www.youtube.com/watch?v=FLZvOKSCkxY&list=PLQVvvaa0QuDf2JswnfiGkliBInZnIC4HL

Thank You