BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Introduction to Data Science

Data wrangling and Feature Engineering

Dealing with Textual Data

Dr. Ramakrishna Dantu

Associate Professor, BITS Pilani

Disclaimer and Acknowledgement

- The content for these slides has been obtained from books and various other source on the Internet
- I here by acknowledge all the contributors for their material and inputs.
- I have provided source information wherever necessary
- I have added and modified the content to suit the requirements of the course

Data wrangling and Feature Engineering – Part-1

- Data cleaning
- Data Aggregation, Sampling,
- Handling Numeric Data
  - Discretization, Binarization
  - Normalization
  - Data Smoothening
- Dealing with textual Data
- Managing Categorical Attributes
  - Transforming Categorical to Numerical Values
  - Encoding techniques
- Feature Engineering
  - Feature Extraction (Dimensionality Reduction)
  - Feature Construction
  - Feature Subset selection
    - Filter methods
    - Wrapper methods
    - Embedded methods
  - Feature Learning
- Case Study involving FE tasks

Dealing with Textual Data

Dealing with Textual Data

Introduction

- If all the data in the world was equivalent to the water on earth, then textual data is like the ocean, making up a majority of the volume (Predictive Analytics, Siegel, 2013)
- Typically, a cleaned and organized table consisting of rows and columns of data is fed as input to an algorithm
- The output from the algorithm is a model that could then be used to predict outcomes from a new data set or to find patterns in data
- Question is…
  - Are the same techniques applicable to extract patterns and predict outcomes when the input data looks like normal written communication?
  - How to extract patterns and discover new knowledge by applying mining techniques, not on ordered data, but on unstructured natural language - text?

Dealing with Textual Data

Introduction

- Text analytics is driven by the need to process natural human language
- Natural language does not exist in a structured format consisting of rows (of examples) and columns (of attributes)
  - Text mining is, therefore, the domain of unstructured data science
- Traditionally, mining refers to the process of separating dirt from valuable metal
- Text mining refers to the process of separating valuable keywords from a mass of other words (or relevant documents) and use them to identify meaningful patterns or make predictions

Dealing with Textual Data

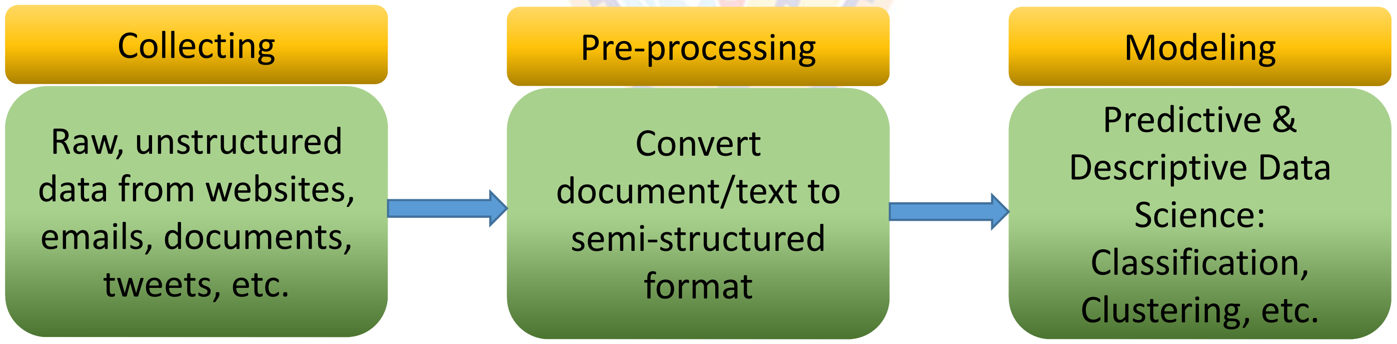
Introduction

- Text mining has many applications, such as:
  - Entity identification
  - Plagiarism detection
  - Topic identification
  - Text clustering
  - Translation
  - Automatic text summarization
  - Fraud detection
  - Spam filtering
  - Sentiment analysis

Dealing with Textual Data

How it works?

- The basic step in text mining involves converting unstructured text into semi-structured data
- Models can be trained on the semi-structured data to detect patterns in the unseen text


Collecting
Raw, unstructured data from websites, emails, documents, tweets, etc.
Pre-processing
Convert document/text to semi-structured format
Modeling
Predictive & Descriptive Data Science: Classification, Clustering, etc.

Dealing with Textual Data

Term Frequency – Inverse Document Frequency

- In a web search, the user types in some keywords and the search engine extracts all the documents (web pages) containing these keywords
- Question is…How does the search engine know which web pages to serve up?
- In addition to using network rank or page rank, the search engine also runs some form of text mining to identify the most relevant web pages
- Suppose that the user types in the following keywords:
  - `"RapidMiner books that describe text mining."`
- In this case, the search engines run on the following basic logic:
  - 1) Give a high weightage to those keywords that are relatively rare
  - 2) Give a high weightage to those web pages that contain a large number of instances of the rare keywords

Source: Data Science – Concepts and Practice by Vijay Kotu and Bala Deshpande

Dealing with Textual Data

Term Frequency – Inverse Document Frequency

- Now, what is considered a rare keyword here?
- Words like "that," "books," "describe," and "text" possibly appear in a large number of web pages
- Whereas "RapidMiner" and "mining" may appear in a relatively smaller number of web pages
- A quick web search for the word (at the time of preparing this PPT)
  - "books" returned about 9,66,00,00,000 results (0.58 seconds), whereas
  - "RapidMiner" returned only about 8,40,000 results (0.41 seconds)
- Therefore,
  - only those pages that not only contain the rare keywords (logic 1),
  - but also have a high number of instances of the rare keywords (logic 2)

  should appear at the top of the search results
- The technique of calculating this weighting is called TF-IDF, which stands for ***term frequency-inverse document frequency***

Dealing with Textual Data

Term Frequency – Inverse Document Frequency

- Term Frequency (TF):

$$TF = \frac{n_k}{n}$$

  - Where,
    - n_k = number of times a keyword (k) appears in a given document
    - n = total number of terms in the document

- Inverse Document Frequency (IDF): is defined as follows

$$IDF = \log_2\left(\frac{N}{N_k}\right)$$

  - N is the number of documents under consideration
    - In a search engine context, N is the number of all the indexed web pages
    - For most text mining problems, N is the number of documents that one is trying to mine
  - N_k is the number of documents that contain the keyword, k

Dealing with Textual Data

Term Frequency – Inverse Document Frequency

- Term Frequency: $TF = \frac{n_k}{n}$
  - a common English word such as "this" will have a fairly <u>high</u> TF score, and
  - a word such as "RapidMiner" will have a much <u>lower</u> TF score
- Inverse Document Frequency: $IDF = \log_2\left(\frac{N}{N_k}\right)$
  - Again, a word such as "this" would appear in every document and, thus, the ratio (N/N_k) would be close to 1
    - Thus, the IDF score would be close to zero
  - However, a word like "RapidMiner" would possibly appear in a relatively fewer number of documents and so the ratio (N/N_k) would be much greater than 1
    - Thus, the IDF score would be high for this less common keyword

Dealing with Textual Data

Term Frequency – Inverse Document Frequency

- TF-IDF is expressed as follows:

$$\frac{n_k}{n} \times \log_2\left(\frac{N}{N_k}\right)$$

  - When the high TF for "this" is multiplied by its corresponding low IDF, a low (or zero) TF-IDF will be reached
  - Whereas, when the low TF for "RapidMiner" is multiplied by its corresponding fairly high IDF, a relatively higher TF-IDF would be obtained
  - Typically, TF-IDF scores for every word in the set of documents is calculated in the preprocessing stage

Terminology
Steps in the Textual Data Processing

Dealing with Textual Data

Terminology – Steps involved in the Textual Data Processing

- Following are the broad steps involved in Text Mining
  - Removing special characters, changing the case (up-casing and down-casing)
  - Tokenization
    - process of discretizing words within a document
  - Creating Document Vector or Term Document Matrix
  - Filtering Stop Words
  - Term Filtering
  - Stemming/Lemmatization
  - Forming n-grams and storing them in the document vector

Dealing with Textual Data

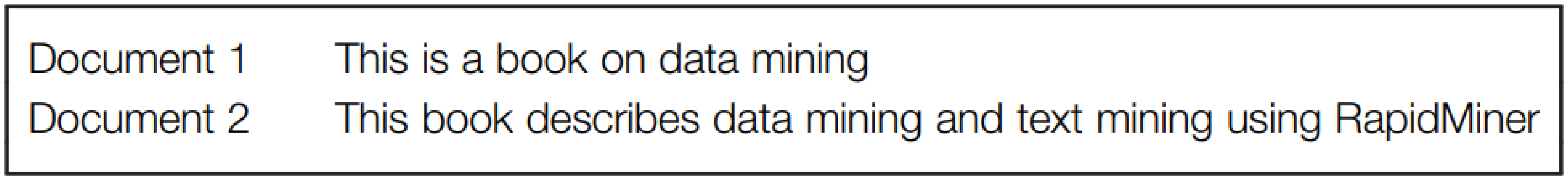
Terminology

- Consider the following two sentences:
  - "This is a book on data mining" and
  - "This book describes data mining and text mining using RapidMiner."
- Suppose the objective is to perform a comparison between them, or a similarity mapping
- For this purpose, each sentence is one unit of text that needs to be analyzed
- These two sentences could be embedded:
  - in an email message, in two separate web pages, in two different text files, or else they could be two sentences in the same text file

Dealing with Textual Data

Terminology - Tokenization

- Document
  - In the text mining context, each sentence is considered a distinct ***document***
- Token
  - Each word is called a ***token***
- Tokenization
  - The process of discretizing words within a document is called ***tokenization***
- A document here is simply a sequential collection of tokens (words)



Document 1	This is a book on data mining
Document 2	This book describes data mining and text mining using RapidMiner

Dealing with Textual Data

Terminology – Document Vector or TDM

- We create a matrix where each column consists of a token and the cells show the counts of the number of times a token appears
- Each token is now an attribute in standard data science parlance and each document is an example (record)
- Basically, unstructured raw data is now transformed into a format that is recognized by machine learning algorithms for training
- This table is referred to as *Document Vector* or *Term Document Matrix (TDM)*

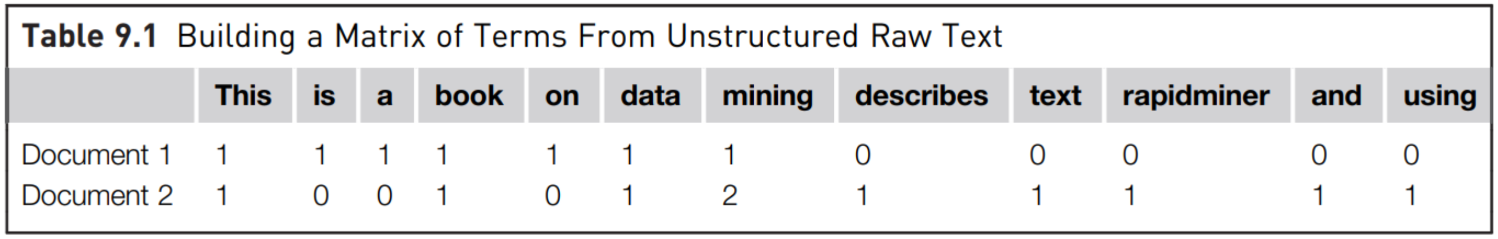


Table 9.1 Building a Matrix of Terms From Unstructured Raw Text

	This	is	a	book	on	data	mining	describes	text	rapidminer	and	using
Document 1	1	1	1	1	1	1	1	0	0	0	0	0
Document 2	1	0	0	1	0	1	2	1	1	1	1	1

Source: Data Science – Concepts and Practice by Vijay Kotu and Bala Deshpande

Dealing with Textual Data

Terminology – Document Vector or TDM

- Suppose a third statement is added
  - "RapidMiner is offered as an open source software program."
- This new document will increase
  - The number of rows of the matrix by one (Document 3)
  - The number of columns by seven (seven new words or tokens were introduced)
- This results in zeroes being recorded in nine other columns for row 3
- As more new statements are added that have little in common, we end up with a very sparse matrix

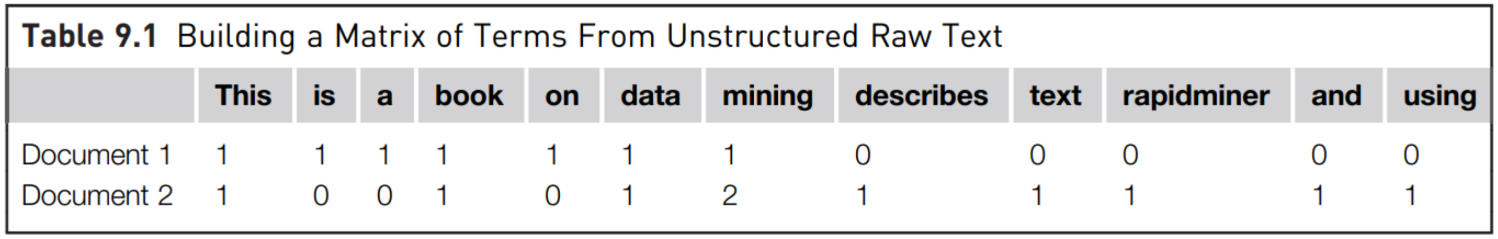


Table 9.1 Building a Matrix of Terms From Unstructured Raw Text

	This	is	a	book	on	data	mining	describes	text	rapidminer	and	using
Document 1	1	1	1	1	1	1	1	0	0	0	0	0
Document 2	1	0	0	1	0	1	2	1	1	1	1	1

Terminology – Document Vector or TDM

- We could also use term frequencies (TF) for each token instead of simply counting the number of occurrences
  - TF can be obtained by dividing number in the table by number of words in the row (document)
- Similarly, TF-IDF scores can also be used for each term to create the document vector

$$TF = \frac{n_k}{n}$$

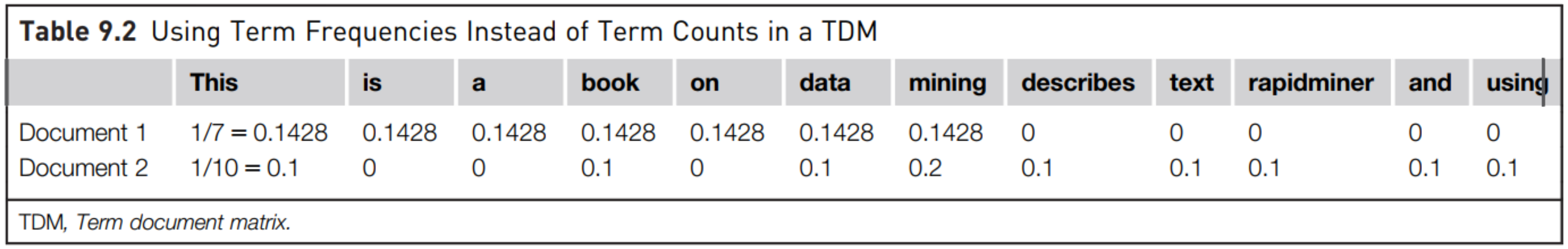
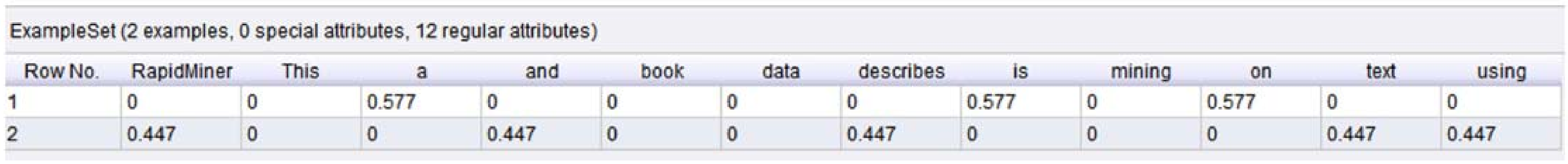


Table 9.2 Using Term Frequencies Instead of Term Counts in a TDM

	This	is	a	book	on	data	mining	describes	text	rapidminer	and	using
Document 1	1/7 = 0.1428	0.1428	0.1428	0.1428	0.1428	0.1428	0.1428	0	0	0	0	0
Document 2	1/10 = 0.1	0	0	0.1	0	0.1	0.2	0.1	0.1	0.1	0.1	0.1

TDM, *Term document matrix.*



ExampleSet (2 examples, 0 special attributes, 12 regular attributes)

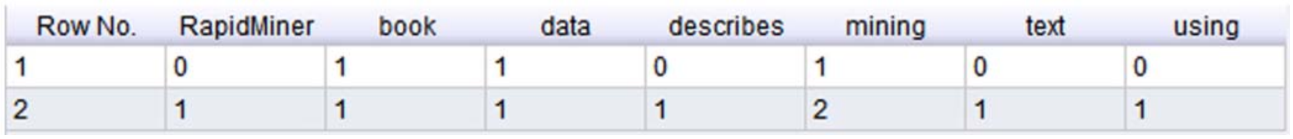
Row No.	RapidMiner	This	a	and	book	data	describes	is	mining	on	text	using
1	0	0	0.577	0	0	0	0	0.577	0	0.577	0	0
2	0.447	0	0	0.447	0	0	0.447	0	0	0	0.447	0.447

Source: Data Science – Concepts and Practice by Vijay Kotu and Bala Deshpande

Dealing with Textual Data

Terminology – Stop Words

- In the sample documents, there are common words such as "a," "this," "and," and other similar terms
- In larger documents, there will be a higher number of such terms that do not really convey specific meaning
- Most parts of speech such as articles, conjunctions, prepositions, and pronouns need to be filtered before additional analysis is performed
- Such terms are called *stop words*
- Stop word filtering is usually the second step that follows immediately after tokenization.
- The document vector gets reduced significantly after applying standard English stop word filtering



Row No.	RapidMiner	book	data	describes	mining	text	using
1	0	1	1	0	1	0	0
2	1	1	1	1	2	1	1

Dealing with Textual Data

Terminology – Stop Words, Lexical Substitution

- In addition to filtering standard stop words, domain specific terms might also need to be filtered out
  - For example, if we are analyzing text related to the automotive industry, we may want to filter out terms common to this industry such as "car," "automobile," "vehicle," and so on
- This is generally achieved by creating a separate dictionary where these context specific terms can be defined and then *term filtering* can be applied to remove them from the data
- *Lexical substitution*
  - It is the process of finding an alternative for a word in the context of a clause
  - It is used to align all the terms to the same term based on the field or subject which is being analyzed
    - This is especially important in areas with specific jargon, e.g., in clinical settings

Dealing with Textual Data

Terminology – Stemming

- *Stemming* is usually the next process step following term filtering
- Words such as "recognized," "recognizable," or "recognition" may be encountered in different usages, but contextually they may all imply the same meaning
- For example,
  - "Einstein is a well-recognized name in physics" or
  - "The physicist went by the easily recognizable name of Einstein" or
  - "Few other physicists have the kind of name recognition that Einstein has."
- The root of all these highlighted words is "recognize."
- The conversion of unstructured text to structured data can be simplified by reducing terms in a document to their basic stems
  - Because now only the occurrence of the root terms has to be taken into account.
- This process is called *stemming*

Dealing with Textual Data

Terminology – Stemming

- The most common stemming technique for text mining in English is the Porter method (Porter, 1980)
- Porter stemming works on a set of rules where the basic idea is to remove and/or replace the suffix of words
- For example:
  - Replace all terms which end in 'ies' by 'y,' such as replacing the term "anomalies" with "anomaly."
  - Stem all terms ending in "s" by removing the "s," as in "algorithms" to "algorithm."
- While the Porter stemmer is extremely efficient, it can make mistakes that could prove costly
- For example:
  - "arms" and "army" would both be stemmed to "arm," which would result in somewhat different contextual meanings

Terminology – Lemmatization

- Lemmatization has this same goal as Stemming, but does so in a more grammatically sensitive way
- For example
  - While both stemming and lemmatization would reduce "cars" to "car," lemmatization can also bring back conjugated verbs to their unconjugated forms such as "are" to "be."
  - Which one to use depends on the case
- Lemmatization uses POS (Part of Speech) Tagging heavily
- POS Tagging is the process of attributing a grammatical label to every part of a sentence

Dealing with Textual Data

Terminology – Lemmatization

- Take the sentence:
  - "Game of Thrones is a television series."
- If we apply POS Tagging on it we get:
  - ({"game":"NN"}, {"of":"IN"}, {"thrones":"NNS"}, {"is":"VBZ"}, {"a":"DT"}, {"television":"NN"}, {"series":"NN"})
- Where:
  - NN = noun, IN = preposition, NNS = noun in its plural form, VBZ = third-person singular verb, and DT = determiner
- POS Tagging is a use case of sentence-tokenization rather than word-tokenization
- After the POS Tagging is complete we can still proceed to word tokenization, but a POS Tagger requires whole sentences
- Combining POS Tagging and lemmatization is likely to give cleaner data than using only a stemmer

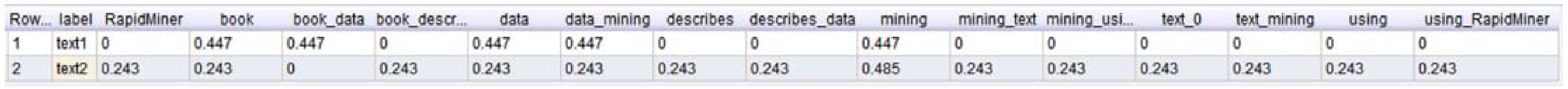
Different techniques for POS Tagging

- Lexical Based Methods
  - Assigns the POS tag the most frequently occurring with a word in the training corpus.
- Rule-Based Methods
  - Assigns POS tags based on rules
  - For example, we can have a rule that says, words ending with "ed" or "ing" must be assigned to a verb
  - Rule-Based Techniques can be used along with Lexical Based approaches to allow POS Tagging of words that are not present in the training corpus but are there in the testing data.
- Probabilistic Methods
  - This method assigns the POS tags based on the probability of a particular tag sequence occurring
  - Conditional Random Fields (CRFs) and Hidden Markov Models (HMMs) are probabilistic approaches to assign a POS Tag
- Deep Learning Methods
  - Recurrent Neural Networks can also be used for POS tagging

Dealing with Textual Data

Terminology – *n-grams*

- There are families of words in the spoken and written language that typically go together
- For example, the word "Good" is usually followed by either "Morning," "Afternoon," "Evening," "Night," or "Day."
- Grouping such terms, called *n-grams*, and analyzing them statistically can present new insights
- The final preprocessing step typically involves forming these n-grams and storing them in the document vector
- Algorithms providing n-grams become computationally expensive and the results become huge so in practice the amount of "n" will vary based on the size of the documents and the corpus
- Table shows a TF-based document vector for bigrams (n = 2) from the examples
- As can be seen, terms like "data mining" and "text mining" and "using RapidMiner" can be quite meaningful in this context

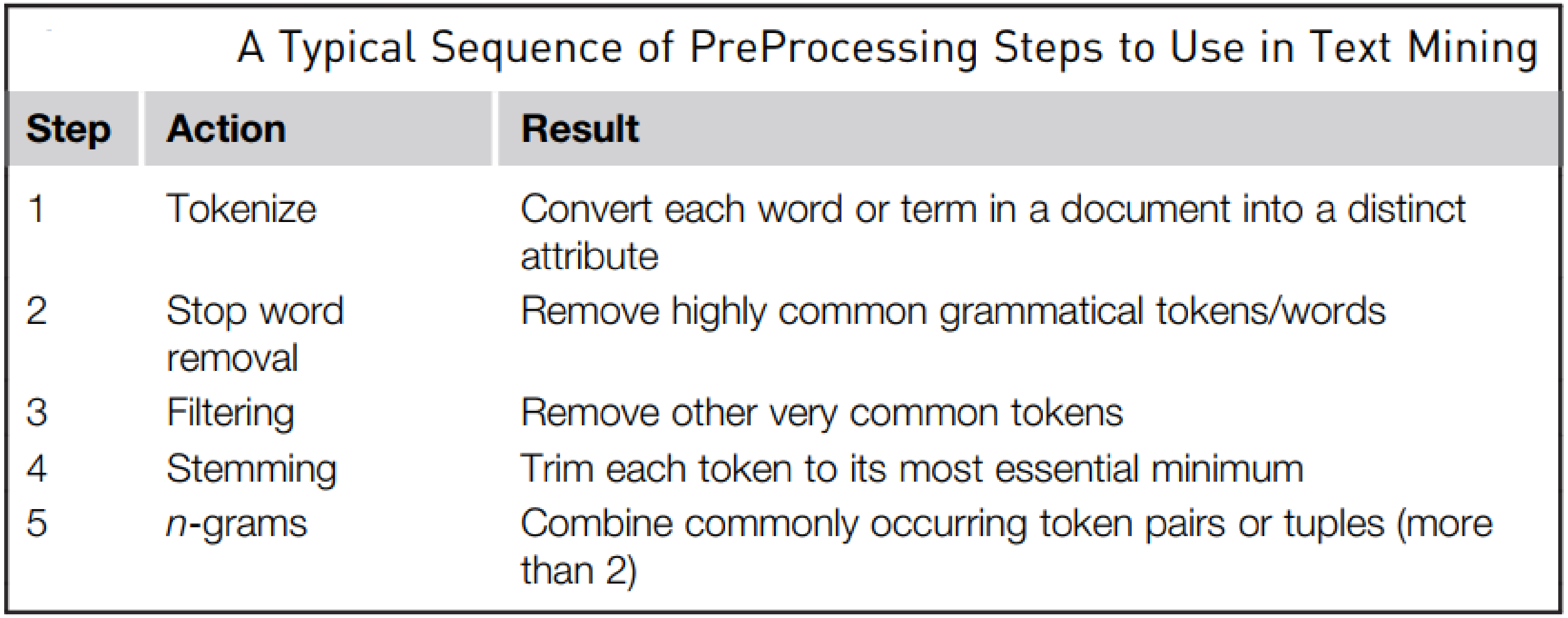


Row...	label	RapidMiner	book	book_data	book_descr...	data	data_mining	describes	describes_data	mining	mining_text	mining_usi...	text_0	text_mining	using	using_RapidMiner
1	text1	0	0.447	0.447	0	0.447	0.447	0	0	0.447	0	0	0	0	0	0
2	text2	0.243	0.243	0	0.243	0.243	0.243	0.243	0.243	0.485	0.243	0.243	0.243	0.243	0.243	0.243

Source: Data Science – Concepts and Practice by Vijay Kotu and Bala Deshpande

Dealing with Textual Data

Summary of Terms and Steps



A Typical Sequence of PreProcessing Steps to Use in Text Mining

Step	Action	Result
1	Tokenize	Convert each word or term in a document into a distinct attribute
2	Stop word removal	Remove highly common grammatical tokens/words
3	Filtering	Remove other very common tokens
4	Stemming	Trim each token to its most essential minimum
5	*n*-grams	Combine commonly occurring token pairs or tuples (more than 2)

Source: Data Science – Concepts and Practice by Vijay Kotu and Bala Deshpande

innovate achieve lead

Thank You!