Text mining to enhance hydroinformatics

Report to the Water Research Commission

by

Yannick Nuapia, Busisiwe Mnguni and Hlanganani Tutu

Molecular Sciences Institute, School of Chemistry, University of the Witwatersrand

WRC Report No. 3074/1/23
ISBN 978-0-6392-0399-7

May 2023

Obtainable from
Water Research Commission
Bloukrans Building, 2nd Floor
Lynnwood Bridge Office Park
4 Daventry Road
Lynnwood Manor
PRETORIA

orders@wrc.org.za or download from www.wrc.org.za

This is the final report of WRC project no. C2021-2022-01091.

EXECUTIVE SUMMARY

BACKGROUND

Hydroinformatics provides various constituents of water-related data, among which is water quality. Most of the sources of such data, which is usually structured in format, have been internal databases of organisations such as water utility companies, private companies and municipalities. This has led to the interpretation of water quality within this ambit and where such data is limited or unavailable, it has impacted on decision making. Text data, unstructured and available on various online platforms (e.g. blogs, chat sites, social media comments, emails, etc), can be an important source of information and one that can augment structured data. This has been widely used in marketing for segmenting and targeting particular customers and, as a way of assessing brand performance. Insurance, security and banking industries have also tapped into this data resource. However, the uptake in water management has been very sparse and limited. The low uptake in water management may be related to traditional perceptions in scientific fields, in general, that data should be structured and in a format to which numerical and deterministic models can be applied. Machine learning techniques have been used with notable success for water data in such formats, but have not been fully explored, as they have been in other fields, for text mining of unstructured water data available on online platforms. Further, the advent of cloud-based systems means that large volumes of such data can be stored together with structured data and made available to a wide variety of users.

AIMS

This study aimed at conducting text mining of water data from Twitter and WhatsApp platforms to enhance hydroinformatics, mainly water quality. This was supported by the following other specific aims or objectives:

- Developing a framework and guidelines, based on coding and a third-party service, to extract text on water quality from online platforms (Twitter and WhatsApp).
- Collating the data from text mining (on Twitter and WhatsApp), transforming it to structured format and modelling it, using machine learning techniques to yield insights into opinions and sentiments.
- To develop, and/or explore from available third-party services, an intermediary database to store and curate extracted information – including establishment of quality control protocols.

To achieve these aims, the approach followed for the research is as follows.

METHODOLOGY

The approach to the study involved: text mining and extraction; processing the text to transform it to structured format; using machine learning techniques to model the opinions and sentiments; and developing a cloud-based database for curating the mined data and models generated from it.

Mining and extracting text from online platforms

Text data from comments on online platforms (Twitter and WhatsApp) were mined and extracted. Only these two platforms were used by virtue of permission that they granted us to use their data. For Twitter, coding using R/RStudio was used to extract data from its platform followed by storing on the Google Cloud platform. This was also attempted using a third-party service provider, Fivetran.

Twitter comments were general, drawn from users commenting on their timelines (and retweeting) about water quality from their service providers in South Africa. No answers were solicited in the form of a survey, but rather just a direct mining of the platform using appropriate search criteria. More than 1500 tweets and retweets were extracted that related to water quality in the country.

For WhatsApp, a group chat was formed of 100 respondents from across the country. They were given a set of questions to respond to with respect to their water quality. They were allowed to give responses for their areas of primary residence, as well as for any other areas that they visit often, e.g. someone living in one district or town or city but working in a different one. The comments were then exported to a spreadsheet prior to processing and modelling.

Opinion and sentiment modelling
Text data was processed, cleaned and transformed from an unstructured to a structured format. The latter is a vectorised form that allows text data to be modelled using machine learning techniques. Analysis of opinions and sentiments was conducted, allowing for delineation of positive, negative and neutral aspects thereof. Different models of word associations/clusters, groupings and networks were created using coding (R/RStudio).

Curation in intermediary database and consideration of quality control aspects
This was conducted through coding and use of a third-party service (Fivetran). For coding, a database was created on Google Cloud in which Twitter comments were stored after they were mined. The models generated for opinions and sentiments drawn from the comments were also stored on the cloud. Storage was also done for WhatsApp comments and models generated from them. A trial version of Fivetran was used to demonstrate the possibility of data storage. Quality control aspects determining data integrity, security and quality were assessed with respect to the use of online platforms and cloud storage.

FINDINGS

The findings revealed that it was possible to mine and extract text data for water quality from Twitter and WhatsApp platforms. The data was successfully processed, cleaned, transformed into structured form through vectorisation and modelled with respect to opinions and sentiments. Informative and interesting patterns were drawn from opinions and sentiments, delineating them to illustrate where respondents or users were satisfied, dissatisfied, trusted or distrusted their water quality. Word clusters, word clouds and networks were obtained that revealed these and several other delineations. This was further corroborated using word connection strengths (to emphasise strong connection paths among words) and polarity distributions that enabled classification of opinions and sentiments as positive, negative or neutral.

Coding was successful in creating a database on Google Cloud for storage of both Twitter and WhatsApp comments and models. Buckets and folders to contain these were created, such that if different teams were to work on a project, the members could access these to make amendments or additions. It was possible to create and store data in a cloud database created with Fivetran. However, the search-and-store approach that was successfully used for coding could not be accomplished with Fivetran, as the source code is based on structured query language (SQL) whose back-end components could not be accessed and changed. SQL works with base tables, i.e. structured tables from which particular data can be accessed using query-based searches. This could not work for data, such as the comments in Twitter, as they are not in the form of base tables, but rather plain, random text. These findings would be true for most third-party services, as their codes are customised and usually based on SQL, and quite widely used in such services.

Quality control of the data, by maintaining data quality, data integrity and data security was identified as important in ensuring that data was not compromised at any stage of the chain of custody. This could not be implemented comprehensively within the scope of the project, as it would have required a substantial time frame and a case study approach. This would ensure elimination of spurious comments from any fake accounts and the weighting of users so as to avoid discrepancies that may be caused by users, known as influencers, who tend to exert more influence on other users.

CONCLUSIONS

The study has shown that text data is important and can be mined, processed and modelled to derive insights related to hydroinformatics in general. This would be important also in instances where other data yields insufficient information about an aspect such as water quality. The study has revealed how useful online platforms are as latent repositories of this data.

Machine learning techniques have also proven their utility, with their ability to model such a complex data structure as unstructured text data through vectorised transformations into structured data. From that, opinions and sentiments were drawn out that revealed important intentions within the comments. Various approaches revealed positive, negative and neutral aspects of these. The implication here is how far such an approach actually involves the communities who are the users of water, rather than drawing inferences only from conventional structured analytical data and surmising from that what the community could be experiencing. Text mining in this case brings out indigenous knowledge capabilities and makes communities involved in revealing the science behind their water, notwithstanding their professional background and level of education.

The creation of an intermediary database using coding proved to be more ideal, as coding is quite versatile and gives more leverage to the user and allows for certain aspects to be created from scratch. This implied that it is possible to create a platform based on coding that would link text mining, processing and modelling, as well as storage, on a cloud platform. The use of coding to extract and curate text data also proved to be a lot more cost-effective compared to use of a third-party service provider. There was less flexibility provided by these providers as they tend to use SQL that would work well with data contained in base tables. As any other component of computing, it is important to protect systems and, subsequently, data from being compromised. This will ensure data quality, data integrity and data quality. There is a shared responsibility here between the cloud providers and users, with the former protecting the underlying cloud infrastructure while the latter protect data and cloud-deployed assets.

RECOMMENDATIONS

The use of text data to augment conventional analytical data is recommended. There is a large host of information that can be drawn from online platforms through text mining of comments and building models based on them. Within this context, it would be recommended that such projects should involve collaboration with researchers in linguistics so that a multi-lingual approach to dealing with online data can be pursued. This will address other aspects within indigenous knowledge systems and citizen science that have become key areas with a potential to enhance scientific research. South African communities are often diverse with multiple languages spoken amongst the people. Tapping into these languages and working through their diversity, extracting information from them, and aligning implied sentiments in them would be useful.

Further, there can be collation of data from diverse sources including conventional data, text data from online platforms, other citizen data such as videos and images, and data from sensors. This will allow for automated collection, modelling and curation of data from various sources.

Extension of this capability of text mining would be recommended to incorporate a wider scope of hydroinformatics beyond water quality. For instance, events such as flooding and climate change can be tracked using text mining. The latter can be assessed by, for instance, having a case study in the form of a village where agricultural activity and its output can be used as indicators.

ACKNOWLEDGEMENTS

The project team thanks the following people and organisations for their contributions to the project.

Name	Affiliation
Shafick Adams	Water Research Commission
Gerda Kruger	Water Research Commission
Sadiya Ooni	WITS University
Environmental Analytical Chemistry Research Group (WITS University), Twitter Inc. and Fivetran	

(Page intentionally left blank)

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS & ABBREVIATIONS

AI	Artificial intelligence
ANN	Artificial neural networks
API	Application Programming Interface
AUC	Area under curve
CTM	Correlated Topic Model
HMM	Hidden Markov Models
IE	Information extraction
IR	Information retrieval
IoT	Internet of things
LDA	Latent Dirichlet Allocation
LSA	Latent semantic analysis
ML	Machine learning
NLP	Natural language processing
PCA	Principal Component Analysis
POS	Parts of speech
R	the statistical programming language developed by **R**oss Ihaka and **R**obert Gentleman (the language derives its name from the inventors' initials)
RStudio	the platform on which the R language is used (an analogy would be speaking in the French language over a telephone (platform))
RMSE	Root mean square error
ROC	Receiver operating curve
TDM	Total document matrix

GLOSSARY

Application Programming Interface: is a software intermediary that allows two applications to communicate with each other. Essentially, it is the messenger that delivers a request to the provider and then delivers the response back to the requester.

Big data analytics: is the use of advanced analytical techniques for very large, diverse data sets that include structured, semi-structured and unstructured data, from different sources, and of different sizes.

Categorisation: is a type of supervised learning where categories are known in advance and is essentially a process of gathering, processing and analysing text documents to put them in the correct category based on the link between the content and category.

Citizen science: the collection and analysis of data by members of the general public. It is sometimes referred to as public participation in scientific research.

Clustering is an unsupervised technique that finds intrinsic structures in information and arranges them into subgroups called clusters to generate labels for objects based on the data.

Hydroinformatics: integrates knowledge and understanding of both water quantity and quality with the latest developments in information technology to improve technical and business decision-making within the water sector. It straddles data capture, storage, processing, analysis and graphical models, and also incorporates the use of modelling, simulation, optimisation and knowledge-based tools and systems infrastructure.

Lemmatisation: is a more efficient process, which uses vocabulary and morphological analysis of words and removes only the inflectional endings to return the base form of word as output.

Stemming: is a process of removing the ends of words, mostly derivational affixes.

Text categorisation: the process of categorising and labelling words into different parts of speeches, also known as tagging.

Text mining: also called text analytics, is an artificial intelligence (AI) technology that uses natural language processing (NLP) to transform the free (unstructured) text in documents and databases into normalised, structured data suitable for analysis by machine learning.

Text summarisation: is the process of automatically creating a compressed version of a given text that provides useful information for the user.

Tokenisation: is the process of breaking up a stream of text, a character sequence or a defined document unit, into phrases, words, symbols, or other meaningful elements.

CHAPTER 1: INTRODUCTION

This chapter presents the background to the study, the motivation, aim and objectives as well as some research questions that the study sought to answer.

1.1 BACKGROUND

Successful water management is hinging increasingly on a combination of sources of data beyond the conventional structured sources, e.g. those provided by water utility companies, municipalities, government departments and agencies. The importance of unstructured data is becoming apparent in general and efforts are being directed at tapping into this type of data. This information and/or data comes from observations by community members, for instance, and are not conventional or structured as is the case with laboratory and scientific field data. This may comprise information gleaned from surveys, interviews, blogs, social media commentary, observations recorded by the community, videos, photos and some data from gadgets (e.g. sensors) for monitoring water and weather events.

An example is a case where a community observes or identifies that water from their borehole is getting fouled over time and shows changes in colour from clear to brownish. This could be associated with levels of iron in the water. Or it can be smells that are sensed in the water, indicating contamination by organic substances. The other common one is the ability or inability of the water to form a lather with laundry soap during the washing of clothes. This is usually associated with elevated levels of alkalinity in the water (Nuapia et al., 2021). This data can be combined with other data, e.g. data from sensors and integrated into mainstream data from laboratory tests to extract optimal value and obtain a comprehensive understanding of water quality.

It is also possible to apply some big data analytical techniques to such data and make some decisions based on that. One such method that can be used is hidden Markov chains in which observed states change (e.g. temporal or spatial changes in the colour of water). These states, that are essentially probability representations, can then be correlated to the underlying prevailing chemistry of the water, e.g. a change in the concentration of iron. The iron concentrations can be further classified using fuzzy logic, a computational intelligence technique based on set theory. Further, the observations can be related to other features such as weather events. For instance, when it rains a lot there could be a change in the quality of water that the community observes. If these are recorded, it is possible to link different parameters and forms of data to obtain a comprehensive picture of the quality of water and to anticipate changes that could potentially occur based on them, making this an early warning system.

1.2 MOTIVATION

As can be observed, most of the data provided by citizens in the methods described above constitutes text data and this can be collated using text mining. Text mining has become an important area of research in deep learning, which is a subset of machine learning and, in turn, of artificial intelligence. It has been used extensively in areas such as advertising, marketing, product reviews, natural language processing (NLP), checks for plagiarism and compiling news feeds. As indicated earlier, some examples of such data include social media feeds (e.g. emails, blogs, Twitter, WhatsApp and Instagram), surveys such as those done using Survey Monkey and general interviews.

Text mining (also referred to as text analysis) is a process of extracting interesting and significant patterns or numeric indices by means of identifying facts, relationships and assertions within textual data (Soumen, 2002). Text mining makes text accessible to various algorithms (e.g. machine learning algorithms) for further analysis (Bruce et al., 2009). According to Manning et al. (2008), it has become an especially important technique in the analytics industry due to the availability of unstructured text

data from multiple sources (vide supra). It is a multi-disciplinary field based on information retrieval, data mining, statistics, and computational linguistics (Nisbet et al., 2009).

Text mining is important owing to the following reasons:

1. it helps in smart decision making and knowledge discovery in different areas,
2. through text mining, data can be visualised in a variety of ways, e.g. graphs, charts and tables,
3. it is easily adaptable to knowledge-driven organisations and projects,
4. it can be used to track opinions over time,
5. text can be indexed and used for predictive analytics,
6. it can be used for summarising large documents and systematic reviewing of literature by pointing out key themes.

Text mining consists of five well-known techniques, namely: information retrieval, information extraction, categorisation, clustering and summarisation.

- Information Retrieval (IR) is a process of extracting relevant information from textual database based on the user's query (Smiley et al., 2009). The most renowned IR systems are Google, Yahoo and other search engines, which recognise those documents on the World Wide Web that are associated to a set of given words (Soumen, 2002).

- Information extraction (IE) is a method of extracting semantic (logic or context) information from textual data by identifying relationships and patterns within semi-structured or unstructured text. IR works best for the extraction of valuable information from structured data and IE for extraction from unstructured data (Nisbet et al., 2009).

- Categorisation is a type of supervised learning where categories are known in advance and is essentially a process of gathering, processing and analysing text documents to put them in the correct category based on the link between the content and category (e.g. spam filtering based on content) (Horto et al., 2003).

- Clustering is an unsupervised technique that finds intrinsic structures in information and arranges them into subgroups called clusters to generate labels for objects based on the data (Lochbaum et al., 2000). Cluster analysis can be used as a standalone text mining tool to achieve data distribution, or as a pre-processing step for other text mining algorithms applied to the detected clusters.

- Text summarisation is the process of automatically creating a compressed version of a given text that provides useful information for the user (Ratinoy and Roth, 2009). A summary is a text that is produced from one or more texts that contains a significant portion of the information, reduced in length and keeps the overall meaning as it is in the original texts (Renders, 2004).

Text mining and analytics are umbrella terms describing a range of technologies for analysing and processing unstructured and semi-structured text data. The unifying theme behind each of these approaches is the need to turn text into a digital form that can be processed by algorithms. Converting text into a structured, numerical format and applying analytical algorithms requires knowledge on how to combine techniques for handling text, ranging from individual words to documents to entire document databases (Smith and Humphreys, 2006). Currently, text mining has not had comprehensive definition because of the disparity of fields from where it comes (Vidhya and Aghila, 2010).

As indicated earlier, text mining is conducted using deep learning algorithms of which recurrent neural networks (RNNs) and convolutional neural networks (CNNs) are some of the most commonly used. The former are applicable for sequential data such as words and sentences and works by identifying important relationships in them and abandoning redundant or non-essential words. RNN is used in the encoder-decoder process which has recently been improved by incorporating the aspect of attention.

Here, even if the words are very far apart in the sentence, the computer may understand their relationship by paying attention to each word in the sentence and gleans a global understanding of that sentence.

Extensions of these tools can be made to include:

1. Automatic image caption generation in which, given an image, the system can generate a caption to describe the contents of the image. This would be important in analysing photos taken by citizens and comparing them to extract important information.
2. Automatic translation of text: is the task where there are sentences of text in one language that should be translated into text in another language. For instance, where citizens have captured information for the same water resource in different languages such as Sesotho, IsiZulu, Afrikaans, etc and the contents have to be compared and grouped.
3. Automatic text classification: is the task of assigning a class label given a text document such as a review, tweet, comment or email. This is typically sentiment analysis.

Following successful text mining of water data from online platforms as discussed above, the storage of this and other related data is possible through the use of cloud-based platforms. This is what has informed and motivated this project, notably: extraction of text data from online platforms (Twitter and WhatsApp in this case), developing models based on machine learning tools and storing the data in an intermediary database on a cloud-based platform. The following outline of the aim, its supporting objectives and research questions help to expound on the motivation and communicate its intended purpose.

1.3 PROJECT AIMS, OBJECTIVES

As the preceding discussions have indicated, traditional methods of water data collection and interpretation do not optimally make use of all the data that can be potentially collected. This owes to their reliance on structured data. Thus, text mining presents an opportunity to harness other possible sources of such data that would otherwise remain largely latent and their value unknown. This study aimed at enhancing hydroinformatics, mainly water quality, using text mining.

To achieve this aim, the following objectives were pursued:

1. To develop a framework and guidelines to extract information on water quality from online platforms (Twitter and WhatsApp).
2. To develop and recommend, where available, suitable tools (e.g. deep learning algorithms) for text data and information extraction from online platforms based on text mining.
3. To collate and extract information from text mining (on Twitter and WhatsApp) and visualising it using a variety of data analytics tools.
4. To develop, or explore from available customised versions, an intermediary database to store and curate extracted information – including establishment of appropriate quality control protocols.

1.4 RESEARCH QUESTIONS

To contextualise the motivation, aim and objectives, the research sought to answer the following questions:

1. Is it possible to extract text data related to water quality from online platforms such as Twitter and WhatsApp?
2. Can the extracted text data be modelled using machine learning techniques to delineate opinions and sentiments on water quality contained in the commentary?
3. Can an intermediary database to store the extracted and modelled text data be developed in the cloud (e.g. Google Cloud) from scratch or using a customised platform?
4. How can quality control aspects be incorporated and ensured for the extraction of text data, the developed models and curation in an intermediary database?

CHAPTER 2: RESEARCH FRAMEWORK APPROACH

This chapter presents an overview of the protocols followed in conducting text mining from online platforms, modelling of text data and storage on cloud platforms. The process flow involved was: extraction of text data from online platforms → building models based on machine learning and deep learning → curation of the data in an intermediary database and consideration of quality control aspects. The chapter gives a general overview of the framework used to achieve the above process, including some theoretical background where necessary to enhance descriptions. Further details that delve into the exact protocols followed are furnished in the respective chapters.

2.1 TEXT MINING PLATFORMS

Text mining followed some generic steps with the ultimate aim to use the outcomes in models to assist in gaining useful insights from the water data. Mining was conducted from Twitter and WhatsApp and the steps conducted were as follows:

- Getting authentication from the social website
- Data visualisation
- Cleaning and pre-processing
- Data modelling using standard algorithms such as opinion mining and clustering
- Anomaly/spam detection, correlations, segmentations and recommendations
- Visualisation of results
- Curation in an intermediary database

The analytic process involving the steps above can be summarised into a workflow (Figure 1).

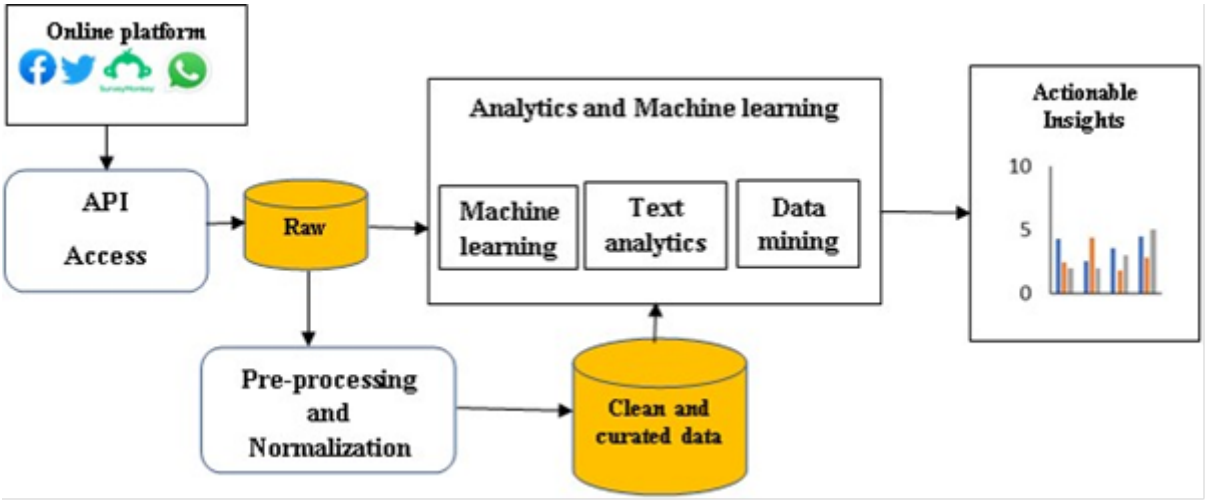

Figure 1 Text mining analytical process workflow

It would be important to note here that authentication for data usage was only obtained from Twitter and WhatsApp and thus the work conducted was limited to these two data sources. However, the procedures described herewith would be the same for any other online data source as the text data tends to be similar and the approaches to conducting mining and analytics not different.

2.1.1 Data access and collection

Getting access from the online platform: OAuth 2.0

Most online platform websites provide an application programming interface (API) access to their data. As a third-party, some mechanism is used to get access to users' data, available on these websites.

Since the user credentials are protected by social media, blog, and web provider due to obvious security reasons, this is where Open Authorisation (OAuth) is used. According to its home page (http://oauth.net/), OAuth can be defined as follows: An open protocol to allow secure authorisation in a simple and standard method from web, mobile and desktop applications. Details of this project were submitted to online platform providers (Twitter and WhatsApp) to allow for authorisation to secure data. This allowed the platforms to provide links to collect data related to the research topic. The obtained raw data was then processed and normalised as needed (Genc-Nayebi et al., 2017).

2.1.2 Data processing

The raw data obtained from data retrieval using online platform APIs may not be structured and clean. In fact, most of the data obtained from online platform is noisy, unstructured, and often contains unnecessary tokens such as Hyper Text Markup Language (HTML) tags and other metadata. Usually, data streams from Facebook, Twitter, blogs, WhatsApp, and other survey APIs have JavaScript Object Notation (JSON) response objects. Some APIs might return data in other formats, such as Extensible Markup Language (XML) or Comma Separated Values (CSV), and each format has to be handled properly. Often data contains unstructured textual data which needs additional text preprocessing and normalisation before it can be used in any standard data mining or machine learning algorithm. Text normalisation was usually done using several techniques to clean and standardise the text. A detailed description of this process was presented in our previous study on citizen science (Nuapia et al., 2021). Most of the techniques described below for working with text, from converting it from unstructured to structured data and modelling are also discussed extensively in that study.

2.1.3 Modelling text data

2.1.3.1 Tokenisation

Tokenisation is the process of breaking up a stream of text, a character sequence or a defined document unit, into phrases, words, symbols, or other meaningful elements called tokens. The goal of tokenisation is to explore words in a sentence. Before doing any kind of analysis on the text using a language processor, the words have to be normalised. When doing quantitative analysis on text, it is considered as a bag-of-words and key words extracted, their frequency of occurrence evaluated, and the importance of each word in the text established. Tokenising provides various kinds of information about text such as the number of words or tokens in a text, the vocabulary or the type of words (Pozzi et al., 2016). The process of tokenisation can be divided into the following operational steps.

Sentence segmentation

Sentence segmentation is the process of determining the longest unit of words. This task involves determining sentence boundaries, and most languages have punctuation to define the end of sentence. Sentence segmentation is also referred as sentence boundary disambiguation or sentence boundary detection. Some of the factors that affect sentence segmentation is language, character set, algorithm, application and data source. Sentences in most of languages are delimited by punctuation marks, but the rules for punctuation can vary dramatically. Sentences and sub sentences are punctuated differently in different languages. Thus for successful sentence segmentation, understanding uses of punctuation in that language is important (Coskun et al., 2018).

Normalising texts

Normalisation in text basically refers to standardisation or canonicalisation of tokens, which were derived from documents in the previous step. Generally, in computer science, this is a process for converting data that has more than one possible representation into a standard, normal, or canonical form. This can be done to compare different representations for equivalence or similarity; to count the number of distinct data structures; to improve the efficiency of various algorithms by eliminating

redundancy in calculations; or to make to introduce a meaningful sorting order. The simplest scenario possible could be a case where query tokens are an exact match to the list of tokens in document, however there can be cases when that may not be true. The intent of normalisation is to have the query and index terms in the same form. For instance, if you query “mining,” you might also be expecting “mining.” Token normalisation can be performed either by implicitly creating equivalence classes or by maintaining the relations between unnormalised tokens (Coskun et al., 2018; Pozzi et al., 2016).

Lemmatisation and stemming

Grammar in every language allows usage of derivationally related words (or derivatives) with similar meaning, which are simply different forms of the same word. Examples are develop, developing, developed. The intent of performing lemmatisation and stemming is based on a similar objective of reducing inflectional forms and map derived words to the common base form.

Stemming is a process of removing the ends of words, mostly derivational affixes. Lemmatisation is a more efficient process, which uses vocabulary and morphological analysis of words and removes only the inflectional endings to return the base form of word as output (Gamal et al., 2019). Derivational and inflectional affixes may have irregular meaning. For instance, consider an inflectional affix such as the plural “s” in word-forms like bicycles, dogs, cars, cats, trees, etc. The difference in meaning between their base form and the affixed form is always the same and points to one commonality of their reference: “more than one.”

2.1.3.2 Categorising tagging text

In corpus linguistics, text categorisation or tagging into various word classes or lexical categories is the second step in the text mining process after tokenisation. Speech parts such as nouns, pronouns, verbs and adjectives play an important role in language. These word classes are not just the salient pillars of grammar, but also quite pivotal in many language processing activities. The process of categorising and labelling words into different parts of speeches is known as parts of speech (POS) tagging or simply tagging. Some of the tagging text approaches to be explored in this project are discussed as follows.

Hidden Markov Models for parts of speech tagging

Hidden Markov Models (HMM) are conducive for solving classification problems with generative sequences. In text mining processing, HMM can be used for phrase chunking and information extraction from comments. If words are considered as input, while any prior information on the input can be considered as states and estimated conditional probabilities can be considered as the output, then POS tagging can be categorised as a typical sequence classification problem that can be solved using HMM (Coskun et al., 2018).

There are five elements needed to define an HMM:

- *N* denotes the number of states (which are hidden) in the model. For parts of text tagging, *N* is the number of tags that can be used by the system. Each possible tag for the system corresponds to one state of the HMM. The possible interconnections of individual states are denoted by *S = {S1, Sn}*. Let *qt* denote the state at time *t*.

- Let *M* denote the number of distinct output symbols in the alphabet of the HMM. For parts of speech tagging, *M* is the number of words in the lexicon of the system. Let *V = {v1 . vm}* denote the set of observation symbols.

- The state transition probability distribution is also called the transition matrix: *A = {aij}*, representing the probability of going from state *Si* to *Sj*. For parts of speech tagging, the states

represent the tags, so *aij* is the probability that the model will move from tag *ti* to *tj*. This probability can be estimated using data from a training corpus:

$$a_{ij} = P\left[q_{t+1} = S_j \mid q_t = S_i\right] 1 \le i, j \le N \quad [1]$$

where: *qt* denotes the current state, the transition probabilities should also satisfy the normal stochastic constraints:

$$a_{ij} > 0, 1 \le i, j \le N \quad [2]$$

And:

$$\sum_{j=1}^{N} a_{ij} = 1, 1 \le i, j \le N \quad [3]$$

- An observation symbol probability distribution is also called emission matrix *B = {bj(k)}*, indicating the probability of the emission of symbol V_k when the system state is *Sj*:

$$b_j(k) = P\left\{o_t = v_k \mid q_t = s_j\right\}, 1 \le j \le N, 1 \le k \le M \quad [4]$$

where: V_K denotes the kth observation symbol O_t and the current parameter vector, the following conditions must be satisfied:

$$b_j(k) \ge 0, \le j \le N, 1 \le k \le M \quad [5]$$

And:

$$\sum_{k=1}^{M} b_j(k) = 1, 1 \le j \le N \quad [6]$$

The initial state probability distribution represents: $\pi = \{(\pi_i)\} \pi = \{\pi_i\}$ probabilities of initial states. For parts of speech tagging, this is the probability that the sentence will begin:

$$\pi_i = P\left[q_1 = S_i\right] 1 \le i \le N, \pi_i \ge 0 \sum_{k=1}^{N} \pi_i = 1 \quad [7]$$

Implementing HMMs

When implementing an HMM, floating-point underflow is a significant problem. When the Viterbi or forward algorithms are applied to long sequences, the resultant probability values are very small, which can underflow on most machines. This problem is solved differently for each algorithm.

Viterbi underflow

As the Viterbi algorithm only multiplies probabilities, a simple solution to underflow is to log all the probability values and then add values instead of multiplying. In fact, if all the values in the model matrices *(A, B, π)* are logged, then at runtime only addition operations are needed.

Forward algorithm underflow

The forward algorithm sums probability values, so it is not a viable solution to log the values in order to avoid underflow. The most common solution to this problem is to use scaling coefficients that keep the probability values in the dynamic range of the machine, and that are dependent only on *t.*

2.1.3.3 Text Summarisation Clustering

High dimensional unstructured data comes with challenges of organising, querying, and information retrieval. The text summarisation and clustering approach is used to extract hidden themes from documents and collections to be able to effectively use it for dozens of purposes such as corpus summarisation, document organisation, document classification, taxonomy generation of web documents, organising search engine query results, news or article recommendation systems, and duplicate content detection (Abuhay et al., 2018).

Topic modelling

Topic models is used for discovering the underlying themes or topics that are present in an unstructured collection of documents. The collection of documents can be organised based on the discovered topics, so that it is easy to browse through the document. There are various topic modelling algorithms that can be applied to a collection of documents to achieve this. Clustering is an extremely popular technique used to group documents, but this does not always fit the requirements. When a text document is clustered, the results in each text exclusively belong to exactly one cluster. In this study, two algorithms were explored, namely: the Latent Dirichlet Allocation (LDA) and Correlated Topic Model (CTM) (Jelodar et al., 2019). This is an intrinsic process, as these algorithms are in the libraries used.

Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) is one of the most widely used topic modelling methods and belongs to a class of models that are called generative models. There are latent themes present in every document, and each word in the document contributes to the theme or topic, which encodes some assumption about the document or collection. By effectively grouping documents with similar underlying topics and themes, it is possible solve trivial issues in searching, organising, and summarising huge archives of unstructured data. Latent topics can be uncovered pervading the collection of documents and annotate the documents according to the topics discovered, which is utilised to extract context, summarise, or organise the collection. The idea behind LDA is the assumption of a fixed number of topics are distributed over the documents in the whole collection (Jelodar et al., 2019). Each document is an amalgamation of multiple topics across the corpus; each topic is an assortment of thousands of words, while each word is an entity that contributes to the theme of the document. Still, a document can only be observed as a whole with everything else hidden. Probabilistic models of topic modelling have the objective to dissect documents to extract those latent features, which can help summarise a document or organise a group of them (Jelodar et al., 2019).

The strategy for extracting themes from a collection of documents is as follows:

- Every word in each document is assigned a topic
- The proportion of each unique topic is estimated for every document
- For every corpus, the topic distribution is explored

The topic labelled to an observed word depends on a *posterior*, which takes into account the topic and proportion parameters defined, and the assignment of topics to each word in a document, as well as to each document in a corpus. The topic can be assumed to be a probability distribution across the multitude of words, while the topic models are nothing but a probabilistic relationship between the latent unobserved themes and fraction of observed linguistic variables. LDA is a model that uses this process.

This model randomly generates observable data values based on some hidden parameters and follows a generative process; in this process a joint probability distribution of all the variables was applied. The probability weights for words were calculated and create the topics based on the weight of each word; with each topic assigning different weights to different words. For this model, the order of the words

does not matter as it will treat each document as a bag of words, this assumption may not be the best, since the sequence of the words in the sentence is lost. The order of the documents also does not matter. This type of language simplification is very rudimentary and often works because it still helps us to understand the semantics of the topics; knowing which words were used in a document and their frequencies makes it good enough to make decisions on which topic they belong to (Jelodar et al., 2019; Alghamdi et al., 2015).

Correlated topic model

In correlated topic model (CTM), collections were used for better understanding of the correlation between the hidden topics in the collection of documents. This technique is useful in instances where the relationship between each topic has to be established and the outcomes used to build graphs about the topics or build a topic of interest or document browser. This will help in navigating through the collection of documents by their topics of interest or preference and make it easy to find the right content from a huge set of documents. An LDA model sets the basic principles for topic modelling and correlated topic modelling is an extension of this, building upon the LDA model. As explained, LDA model does not consider the order in which the words occur or whether the order of the words is lost or is exchangeable. LDA is a high dimensional vector model that makes an assumption that the occurrence of one topic is not correlated to another topic. For example, an LDA fails to directly model correlation between topics, whereas a correlated topic model (CTM) is a hierarchical model and provides better insights about the data, resulting in better visualisation. A CTM models the words of each document from document-specific random variables and captures the diversity in grouped data that illustrates multiple hidden patterns (Alghamdi et al., 2015). A CTM gives better predictive performance but comes at the expense of extra computation cost.

For both models; LDA and CTM, the number of topics must be fixed before modelling a corpus and that follows a generative process:

- Determine term distribution for each topic
- Determine proportions of the topic distribution for the document
- For each word choose a topic, then choose a word conditioned on that topic

For topic modelling, the following steps were followed:

1. Provided data can be in various formats. Creating a corpus or vocabulary out of the given data is the first step.
2. Process the created corpus to remove noisy data. This involves:
    - Tokenising
    - Stemming
    - Stop word removal
    - Removing numbers
    - Removing punctuation
    - Removing terms below certain length
    - Converting to lower case
3. Create the document term matrix of the processed corpus.
4. Remove the sparse entries from the document term matrix.
5. This matrix can be provided as the input to LDA and CTM

Model selection

Selecting the number of topics presents some challenges. For instance, to fit a given document-term matrix, using the LDA model or the CTM, the number of topics needs to be fixed before modelling.

There are various approaches to select the number of topics (Ju et al., 2012). In this study, the following were used as provided in the libraries in RStudio:

- Bayesian approach
- Hierarchical Dirichlet process
- Cross validation on likelihood approach

Text classification

Text classification is an extensively used phenomenon in text processing and has widespread utility in different domains. Also known as text categorisation, text classification finds its usage in various tasks related to information retrieval and management. Spam detection in e-mails, opinion mining or sentiment analysis on social media data, priority e-mail sorting, intent identification from user queries in chatbots, and automated query answering mechanisms are a few examples where text categorisation has proved to be highly effective. There are several algorithms used for text classification. In this study, inductive learning and tree-based learning algorithms were used (Coskun et al., 2018). These are algorithms that are intrinsic, as they are included in the libraries used.

Inductive learning

Classification or the supervised learning mechanism in machine learning, is the process of learning concepts or patterns generalising the relationship between the dependent and independent variables, given a labelled or annotated data. A typical text classification task can be defined as follows:

- **Task T**: to classify opinions (positive or negative)
- **Performance measure P**: percentage of opinions correctly classified
- **Training experience E**: Annotated or labelled data to train the model on

For an opinion classification, the training data contains texts as instances and opinions (positive or negative) as the outcome variable. The objective is to design a learning mechanism to be able to utilise the patterns in the training data to predict/label the outcome variable in an unknown or test dataset.

To learn the target concept or pattern, each instance *t* along with the associated concept *c(t)* from the training set is presented to the learner. The task assigned to the learning mechanism is to estimate the function *c*, such that the target concept stays generalised over most of the training instances and can be applied on unknown instances with high precision. The classifier creates a hypothesis for every training instance presented to it in conjunction with the associated label for the given instance. Once all the instances are observed, a large set of hypotheses is generated, which are extremely specific in nature.

With the inductive learning hypothesis principle, if a hypothesis can effectively approximate the target concept over a sufficiently large number of instances, it will also effectively approximate over an unknown set of instances. Such a hypothesis needs to be a generalised one as a specific hypothesis can approximate over a sufficiently large of number of instances and it would prove to be insufficient to approximate over unobserved instances (Pozzi et al., 2016).

Tree-based learning

In machine learning, the decision tree is a well-known classification algorithm. In this type of classifying methodology, a decision tree model is created. When an input is provided for a prediction based on the input variable, it traverses through the nodes and reaches the leaf node which is a classifier class. When the target variables have a finite set of values, it is called a classification tree. If the target variables take continuous values, it is called a regression tree (Gamal et al., 2019).

A decision tree is constructed based on input variables. The number of input variables will alter the classification output, these input variables are also called attributes. In text mining, when classifying a set of documents into different topics, the significant words in the document become the attributes and the topics which are the resulting outcome become the classes.

The tree is made up of branches and nodes. Every branch from a node signifies the outcome using one of the attributes. The nodes that do not have any branches are called leaf nodes. These nodes are at the end of the tree, and they are the final classes of the classification problem. Decision trees are built by recursively splitting training data so that each division tends towards one class. Every node that is not a leaf node has a branch and each branch is the outcome of one or more attributes which further influences how the data will be divided further. The split is made in such a way that the data is distinct or as distinct as possible. In machine learning, this is called a pure leaf node; each division is either pure or we can improve the performance by increasing the generalisation by pruning the tree.

Partitioning of data for splitting the node depends on the attributes used to split it. To construct the tree, we need to select the best splitting attribute. There are various algorithms used to build a decision tree. At a high level, these algorithms try to solve the challenges in their own optimal way as explained in the following steps (Pozzi et al., 2016; Alghamdi et al., 2015):

1. Select the best attributes for splitting and determine the split values.
2. Ascertain the number of splits at each node.
3. Ascertain the order of the attribute that has to be considered for splitting.
4. Choose the pruning method for the tree: pre pruning or post-pruning.
5. Choose the growth and stopping criteria for the tree.

In order to perform an optimal split and evaluate the goodness of the split, there are various methods:

- *Gini index* - is calculated by subtracting the sum of squared probabilities of each class from one. It favours larger partitions and is easy to implement. A feature with a lower Gini index is chosen for a split.
- *Information gain* - is the reduction in entropy or surprise by transforming a dataset and is often used in training decision trees. Information gain is calculated by comparing the entropy of the dataset before and after a transformation. It favours smaller partitions with distinct values compared to the Gini index. Information Gain = entropy(parent) – [average entropy(children)]
- *Entropy* - Entropy is the average rate at which information is produced by a stochastic or random source of data. It controls how a Decision Tree decides to split the data. It actually effects how a Decision Tree draws its boundaries. Gini Index has values inside the interval [0, 0.5] whereas the interval of the Entropy is [0, 1].
- *Gain ratio* - is modification of information gain that reduces its bias. Gain ratio overcomes the problem with information gain by taking into account the number of branches that would result before making the split. It corrects information gain by taking the intrinsic information of a split into account.

2.2 CURATION IN INTERMEDIARY DATABASE

The possibility to use available platforms (e.g. cloud based) was explored using both construction from scratch using coding in R and RStudio as well as using a third party or customised platform. For the latter, a demonstration version of Fivetran was used. There are a number of such resources available now owing to different product offerings available and they generally perform most functions that a user may require.

Detailed descriptions of how this was achieved are furnished in chapter 4.

CHAPTER 3: TEXT DATA MODELLING

This chapter presents the findings related to modelling of text data collected from the Twitter and WhatsApp platforms. The report describes the development of text mining tools and approaches based on different algorithms for analysing text data from these platforms. The workflow comprised the following steps:

- *Data setup, including registration*
- *Connecting to Twitter via R and searching for tweets. For WhatsApp, comments were collated separately*
- *Data processing and clean-up*
- *Opinion modelling*
- *Sentiment analysis*

3.1 TEXT DATA MODELLING: USING TWITTER COMMENTS

The Twitter platform offers some interesting opportunities for text mining. Amongst the platforms considered for the study, it was the most accessible with respect to permission and authentication granted. Notwithstanding, Twitter offers attractive data structures in that the words in tweets are limited (up to 280 characters or Unicode glyphs) and, more often, users aim to get to the points of their conversations a lot more precisely. This makes it easier to search for words and phrases. Text mining has been shown to be useful in extracting information from such words and phrases by modelling the latent sentiments that they contain, thus augmenting structured data.

3.1.1 Classification of opinions from tweets

3.1.1.1 Data setup

Loading dataset from Twitter

Tweets are more than just messages. Each tweet contains some metadata, which is essentially data derived from other data. Metadata can be helpful in understanding or attaching meaning to each data point. In case of a tweet, its date-time, its source and the number of retweets is what makes up the metadata. This additional information and other attributes about each tweet helps us draw various insights. Thus, these were taken into account in establishing datasets on water quality comments drawn from Twitter.

Application Programming Interfaces

Twitter's Application Programming Interfaces (APIs) are the gateway to the immense data of Twitter. These provide some useful utilities for interacting with Twitter in a programmatic way. APIs may be used to develop third party apps along with a way to extract data for analysis/research purposes. For this work, aspects of APIs and objects were studied and used for analysis. Some of these included follower relationships and velocity.

To enable faster development and hassle-free access, Twitter has libraries available in all major programming languages. For this study, the Twitter package for R was used. This package provides ways to connect and extract information from Twitter quite easily.

Registering an application

The first step is to register as a user and then create an app. Twitter requires users to create an app to use its APIs. It uses **Open Authentication** or **OAuth** to grant access to its APIs and data under certain terms and conditions. An app can be easily created by following these steps:

1. Log in to the app management console at http://apps.twitter.com.
2. Once logged in, click on the **Create New App** button. Fill up the required fields. A callback URL is then created. The callback URL is the address that Twitter's API replies to with a response to the queries.
3. Click on **Create Your Twitter Application** to complete the process. Once done, the user is redirected to the app details page which will contain the required details for connecting to it.

The app creation process provides OAuth parameters such as the API Key, API Token, Access Token, and Access Token Secret. Together, these four parameters are useful for tapping into Twitter's APIs for the required cases.

Connecting to Twitter using R

After securing the OAuth parameters and credentials, the package “rtweet” was downloaded in R which is used for extracting tweets. Within the “Source” or “Console” in R, the following commands were executed:

```
install.packages("rtweet")
library (rtweet)
```

Authentification was conducted to connect to Twitter. This was done by entering the name of the app (e.g. Water Research in this case), consumer key and consumer secret - all of which was information received when applying for the Twitter API. Once authentication was verified and accepted, connecting the app to OAuth was conducted using the following code snippet:

```
# load the package

library(twitteR)
library(httr)
devtools::install_github("***/rtweet")
library(rtweet)
consumerKey <- "(XXXXXXXXXXXXXXX)"
consumerSecret <- "(XXXXXXXXXXXXXXXX)"
accessToken <- "(XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX]"
accessTokenSecret <- "(XXXXXXXXXXXXXXXXXXXXXXX)"
setup_twitter_oauth(consumerKey, consumerSecret, accessToken, accessTokenSecret)
```

Search Twitter for Tweets

After setting the above parameters, recent tweets could be searched. The search was conducted for tweets with “#Water comments in South Africa.” The rtweet::search_tweets() function was used to do the search. search_tweets() requires the following arguments:

```
## search for 1000 tweets using the # water comments South Africa
water <-search_tweets("water  comments  South Africa,” n=1000, include_rts=TRUE, lang= "en")
```

Various arguments can be used to achieve different objectives in the search criteria. Some common ones and their meanings are presented (Table 1). The first three have been used in the code snippet above.

Table 1 Some arguments used in tweet searches with meanings

Argument	Meaning
n	Integer, specifying the total number of desired tweets to return. Defaults to 100. Maximum number of tweets returned from a single token is 18000.
include_rts	Logical, indicating whether to include retweets in search results. Retweets are classified as any tweet generated by Twitter's built-in "retweet" (recycle arrows) function. These are distinct from quotes (retweets with additional text provided from sender).
lang	Language of search, e.g. “en” for English. This will return only searches in English.
q	Query to be searched, used to filter and select tweets to return from Twitter's REST API. It must be a character string not to exceed maximum of 500 characters. Spaces behave like a Boolean "AND" operator. To search for tweets containing at least one of multiple possible terms, separate each search term with spaces and "OR" (in caps). For example, the search q = "water quality" looks for tweets containing both "water" and "quality" located anywhere in the tweets and in any order. When "OR" is entered between search terms, query = "water OR quality,” Twitter's REST API should return any tweet that contains either "water" or "quality.” It is also possible to search for exact phrases using double quotes. To do this, either wrap single quotes around a search query using double quotes, e.g. q = '"water quality"' or use a single backslash, e.g. q = "\"water quality\".” Filters can also be used to enhance queries, e.g. retweets and quotes can be excluded by using "-filter:retweets" and "-filter:quotes,” respectively.
type	Character string specifying which type of search results to return from Twitter's REST API. The default is type = "recent,” but other valid types include type = "mixed" and type = "popular.”
retryonratelimit	Logical argument indicating whether to wait and retry when rate is limited. This argument is only relevant if the desired return (n) exceeds the remaining limit of available requests (assuming no other searches have been conducted in the past 15 minutes, this limit is 18000 tweets). Defaults to FALSE. It is set to TRUE to automate the process of conducting big searches (i.e. n > 18000). For many search queries, e.g. for specialised or specific searches, there would not be more than 18000 tweets to return. But for broad, generic, or popular topics, the total number of tweets within the API window of time (7-10 days) can easily reach the millions.
verbose	Logical, indicating whether or not to include output processing/retrieval messages. Defaults to TRUE. For larger searches, messages include rough estimates for time remaining between searches.

3.1.1.2 Data processing and cleanup

The vocabulary matching approach followed the RTextTools method which was used to create a matrix function. The text2vec package is applied to create a vocabulary and training of a document text matrix (DTM). The vocabulary was applied to construct a train set DTM and also the new survey data. The text2vec’s text organisation functions are illustrated to show another text organisation method that can be used for machine learning. The caret (**C**lassification **A**nd **RE**gression **T**raining) package was used for data preparation and the tm package added for preprocessing functions. The glmnet library was loaded to fit the elastic net model for regression, including logistic and multinomial regression using coordinate descent. Finally, the pROC library (for the receiver operating curve) was used to help in visualising model performance. Their libraries were loaded as follows:

```
library(text2vec)
library(caret)
library(tm)
library(glmnet)
library(pROC)
```

The code below is an example of a custom pre-processing function. It helps to reduce errors and make the code more concise. Here, only two functions were applied.

```
data.clean<-function(x){
x<-removePunctuation(x)
x<-stripWhitespace(x)
return(x)
}
```

The custom function data.clean is applied to the final text vector. Using caret's createDataPartition, 70% of the records were assigned to train with the remaining patient rows becoming the test set. Essentially, data partitioning is the act of splitting available data into two portions, usually for cross-validation purposes. One portion of the data is used to develop a predictive model and the other to evaluate the model's performance. The code snippet is as follows:

```
train<-createDataPartition(water$text,p=.7,  list=F)
train.water<-water[train,]
test.water<-water[-train,]
```

The text2vec library differs in DTM construction because of the use of an iterator. An iterator is a method of applying functions in an object. The function is applied as the iterator traverses through the object rather than to an entire object. This means that an iterator can apply functions to large objects that may be too large to fit in memory. Here "itoken" iterates through "text." Along the way, "tolower" is applied along with another function "word_tokenizer." The word_tokenizer function wraps str_split to separate individual words. The snippets of how this is coded are shown below:

```
iter.maker<-itoken(train.water$text,  preprocess_function  =  tolower,  tokenizer  =
word_tokenizer)
```

The iterator object is a set of instructions that are passed to another function create_vocabulary. Here, v is a list of unique words and statistics from the "text."

```
v <-create_vocabulary(iter.maker,stopwords=stopwords('en'))
```

After the above steps, the vocabulary object is passed to a "vectorizer." The vectorizer allows for the creation of a corpus object by instructing R to make a vector of terms. It is apparent from here that this is the way the package name "text2vec" came about. Such transformations have also been used, e.g. in image processing where images are defined by "tensors" or vectors.

```
vectorizer <- vocab_vectorizer(v)
```

The vectorizer is needed to construct a DTM using the text2vec package. Another iterator is needed to create the DTM so itoken is used again. The "it" objects along with the original vectorizer is passed to the "create_dtm" function to get a matrix.

After loading the tweets and reviewing the opinion column form, both data sets were converted as.character strings.

```
it <- itoken(train.water$text, preprocess_function = tolower, tokenizer = word_tokenizer)
```

```
dtm <- create_dtm(it, vectorizer)
```

It can be observed here that essentially text data that is unstructured has been converted to a format that is somewhat structured that uses vectors. This was one of the key aspects noted in the report by Nuapia et al. (2021) in which unstructured citizen science (mainly text data collected through community interviews) was merged with structured analytical water data using machine learning techniques.

For purposes of enhanced understanding and follow through, the above steps are explained further and simplified in the following text (Ju et al., 2012; Jelodar et al., 2019). The purpose of an iterator is to learn statistics about text that is too large for in-memory analysis. Thus, an iterator containing a set of instructions is created solely to explore the text's vocabulary. The vocabulary information is held in a list, but must be changed to a vector in order to construct a text matrix, the DTM. The iterator is "vectorized" using "vocab_vectorizer." The first iterator is only used for vocabulary construction. Then a second iterator is needed to traverse through the text again. This second iterator is passed to "create_dtm" with the vectorised vocabulary from the first iterator's analysis.

The need for two iterators is that both are used for different purposes. The first is for vocabulary and the second for the matrix construction (or feature construction). The dtm object is based on simple term frequency. i.e. the number of times a word or phrase appears. When working on other data, there may be a need for Term Frequency-Inverse Document Frequency (TF-IDF) weighting instead. This is a technique to quantify words in a set of documents by computing a score for each word to signify its importance in the document and corpus. However, there was no need to use this technique in this study.

3.1.2 Opinion modelling

To create the object cv, the sparse matrix train.matrix was passed into the cv.glmnet function followed by definition of the outcome or y variable. The sparse matrix contains the word columns but not the 0 or 1 outcome variable. Thus, the y= parameter uses the column train.water$opinion from the original training set. Further, the column contains numbers and must be changed to a categorical variable using as.factor.

As indicated earlier, the glmnet package was used that fits generalised linear and similar models by using penalised maximum likelihood. By definition, maximum likelihood is a statistical method for estimating population parameters (such as the mean and variance) from sample data that selects as estimates those parameter values maximising the probability of obtaining the observed data. The penalisation aspect there relates to the regression method yielding a sequence of models, each associated with specific values for one or more tuning parameters (Alghamdi and Alfalqi, 2015; Brownlee, 2019). The glmnet-package fits lasso and elastic-net model paths for regression, logistic and multinomial regression using coordinate descent (essentially, optimisation of error in a fixed direction). Modelling of opinions on water data involved a series of steps (Genc-Nayebi and Abran, 2017; Gamal et al., 2019; Abuhay et al., 2018).

Below is a snippet of part of the code used and some brief explanations of the process leading to building the model:

```
text.cv<-cv.glmnet(dtm,y=as.factor(train.water$Opinion),alpha=0.9,family='multinomial',
type.measure= 'auc', nfolds=5, intercept=F)
```

The alpha parameter, which is the mixing input for how regularisation (penalty) is conducted, is calculated. As written alpha=1 will ensure that a lasso regression model is created. Lasso regression is a type of linear regression that uses shrinkage. Shrinkage is where data values are shrunk towards a

central point, like the mean and as such the procedure encourages simple, sparse models (i.e. models with fewer parameters).

The “family=” parameter defines how the model should behave. The complete family parameter options include “Gaussian,” “binomial,” “Poisson,” “multinomial,” “cox” and “mgaussian.” Since the outcome is multiple classes, the family is multinomial, forcing the model to build a logistic regression. In logistic regression, the model defines a line and then finds a set of coefficients for the line that best separates the classes. Its solution is not analytical (as that for linear regression), but rather involving searching the space of possible coefficient values using an efficient optimisation algorithm. The “nfolds” input defines the number of cross-validation folds to perform. The higher the number, the longer the computation time, but the better the reliability of the model. The last parameter “type.measure=” class selects the best penalty “lambda,” λ, value from among the cross-validation models based on the lowest misclassification rate between clickbait and legitimate headlines. Other measures can be chosen depending on the model family and the misclassification implications. The “type.measure” inputs include “deviance,” “AUC,” “class,” “mse” and “mae.”

The text.cv object is a list containing the model information. The outcome can be obtained by calling plot on the model object. The graphic demonstrates the relationship between the misclassification rate and lambda (λ) penalties. Changing the “type.measure” parameter when calling cv.glmnet will change the response variable (y axis) accordingly. Within the graph (Figure 2), there are two verticals dotted lines. The far left line represents the lambda value which minimises the misclassification rate. The other dotted vertical line represents the highest regularisation value within one standard deviation of the minimal class error. A model based on the second penalty value will have fewer inputs compared to a model using the first dotted line lambda value. This helps to make an informed decision to balance complexity against accuracy when making predictions. The plot is obtained by using the code:

```
plot(text.cv)
```

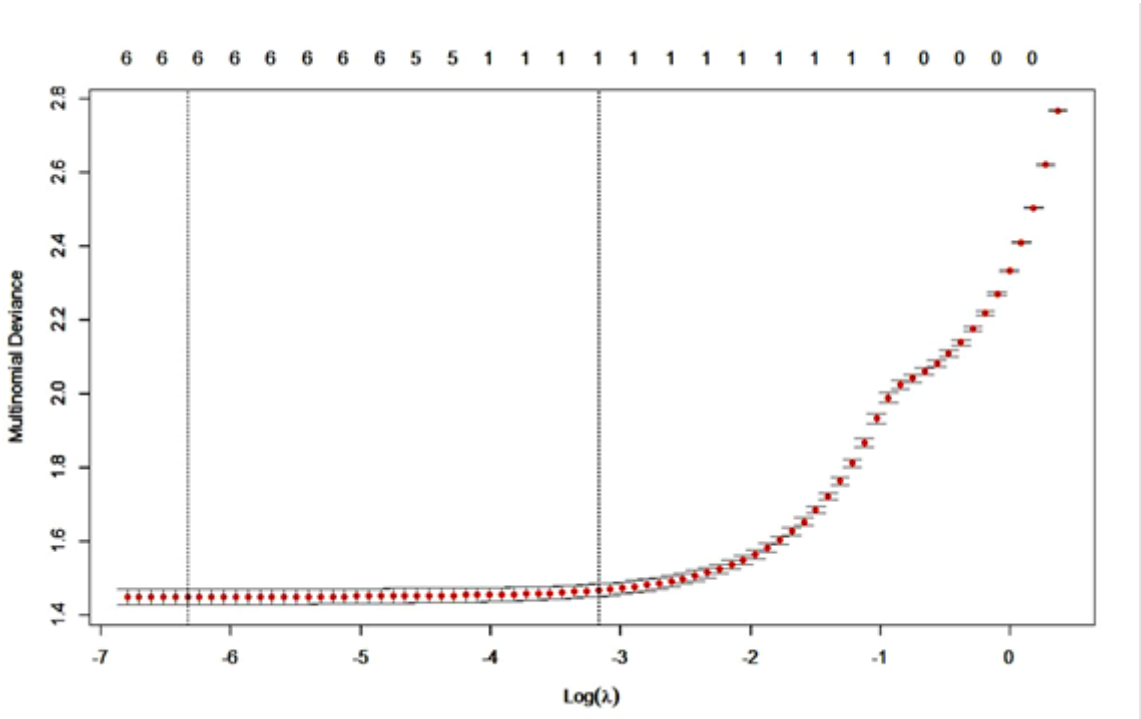

Figure 2 Text-only GLMNet model accuracy results

In addition to the above predictions of model accuracy, the receiver operating characteristic curve (roc) was plotted (not shown here) to determine the sensitivity and specificity of the model. A ROC curve is a plot that illustrates the diagnostic ability of a binary classifier system (true positive and false positive) as its discrimination threshold is varied. The area under the curve (AUC) was also determined as a performance indicator for classification. For a good model, this value should be close to 1.

The code used was as follows:

```
train.auc<-roc(train.water$y,as.numeric(preds))
train.auc
```

An AUC value of 0.92 was obtained, implying that the model was performing well. Intersections of the same classes denote that the model predicted correctly while class differences would denote incorrect prediction. This is determined from a confusion matrix, which is essentially a table that is used to define the performance of a classification algorithm. It was constructed as follows from the code:

```
confusion<-table(preds,train.water$y)
```

Out of a test set of 210 tweets, 198 were found to be correctly classified while 12 were incorrectly classified. To calculate overall model accuracy, sum the diagonal, representing the correct predictions for both classes, then divide by all observations. The code below uses the diag function nested inside the sum function and then divides that amount by the matrix sum. Diag will extract all values along the diagonal, no matter what the size of the matrix. This vector is then summed. In this case, the model classified 94.29% (198/210) of tweets correctly.

```
sum(diag(confusion))/sum(confusion)
```

With the cross-validation done, it was then possible to conduct predictions on new tweets. This was done by establishing a matrix containing values with the same weighting as the training data, along with matching column names. The “data.clean” function was then applied to the test set to ensure preprocessing consistency. Then the “match.matrix” function was applied to the test tweets. The same weighting for headline objects was used. The function “train.dtm” ensured that the columns match between the training and test matrices. Any test words that did not appear in the training set were dropped. The words in common were retained and columns of zeros appended for any words that were in the training set but not in the test set. This is done because it is impossible for the lasso regression to make a classification on information that it has not already seen, such as new words in the test set. Further, the regression model is expecting to have a value for all columns presented during training, thus zeros are used for this purpose where values would otherwise be missing. The code used is as follows:

```
clean.test<-data.clean(test.water$text)
test.dtm<-match.matrix(clean.test,
weighting=tm::weightTfIdf,
original.matrix=train.dtm)
```

The previous train.dtm and test.dtm were compared by calling the objects in the RStudio console. The code below calls both test.dtm and train.dtm:

```
test.matrix<-as.matrix(test.dtm)
test.matrix<-Matrix(test.matrix)
```

To make predictions on the test data, the predict function was used with the model object. The predict function accepts the cross-validation (cv) model first. Then the response type “class” is specified. With “class” the class predictions, 1 or 0, are returned. In contrast, using type ="response" will produce the probability for each of the test tweets. The first represented the lambda value that minimised the overall misclassification error. The code below specifies the minimum lambda value from the cross validated model. That value was captured as cv$lambda.min. To reduce inputs, improve model simplification and retention of model accuracy, the specification, s=cv$lambda.1se was used.

```
preds<-predict (cv,test.matrix,type='class', s=cv$lambda.min)
tweet.preds<-data.frame(doc_row = rownames(test.water),class=preds[,1])
```

The second line creates a concise data frame with appropriate column names and references the test rows from the original data set. Class is a categorical factor and should be changed to numeric for calculating the ROC (Table 2).

Table 2 Sample of first seven rows from tweet.preds

Observed class	Predicted class
1	1
1	1
1	1
1	1
1	1
1	1
1	0

Test Set Evaluation

The preds object is a matrix with just over 1500 results. Evaluation metrics can be calculated on the test set because the test data represents known tweets and outcomes. When the model is applied to new headlines, outcomes are unknown and so calculating the ROC is not possible. Thus, practically, periodic sampling and classification reviews help to keep the model from becoming irrelevant. With capabilities of creating models in the cloud, this is becoming more important as mining and evaluation of tweets and comments can be conducted in real time.

3.1.3 Sentiment analysis

To bring everything together in text mining in this context, the tones, intents and emotions behind the tweets have to be established. This is usually achieved using clustering tools. Sentiments can be generally classified as positive, neutral, or negative (Figure 3). They can also be represented on a numeric scale, to better express the degree of positive or negative strength of the sentiment contained in a body of text (Coskun and Ozturan, 2018; Pozzi et al., 2016). Further, they can be separated into specific sentiments, e.g. fear, joy, anticipation, etc. The Syuzhet package was applied to generate sentiment scores. The package extracts sentiment and sentiment-derived plot arcs from text using a variety of conveniently packaged sentiment dictionaries. This can be summed as follows:

```
library(syuzhet)
    tweets <- iconv(tweet$text)
    s <- get_nrc_sentiment(tweets)
    barplot(colSums(s), las = 2, col = “red,” “grey,” “green,  ylab = 'Count',
    main = 'Sentiment Scores for water -Tweets')
```

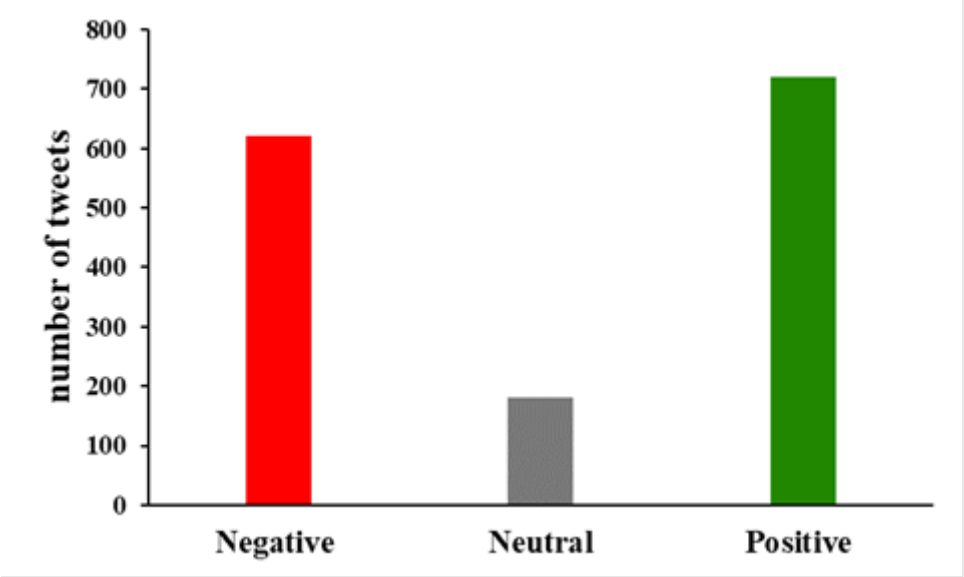

Figure 3 Sentiments from classification of tweets (n = 1520)

The figure summarises the number of tweets with their general opinions on water quality. For a total of 1520 tweets collected, 720 tweets have commented to be happy with their quality of water (classified as positive), 620 tweets have indicated unhappiness with the quality of their water (classified as negative) while 180 tweets did not share positive or negative comments on their water quality (classified as neutral).

To conclude, the above tools and approaches have pointed to how feature construction was conducted using text drawn from Twitter data. This is easily the core and backbone of text mining as it involves conversion of unstructured textual data into a feature-based representation. Since traditional machine learning and data mining techniques are generally not designed to deal directly with textual data, feature construction is an important preliminary step in text mining, converting source documents into a representation that a data mining algorithm can then work with.

Lastly, sentiment analysis was conducted based on emotions and opinions insinuated from the tweets to indicate whether the water quality was viewed to be acceptable (positive), unacceptable (negative) or whether there was some indifference (neutral).

3.2 TEXT DATA MODELLING: USING WHATSAPP COMMENTS

The above discussion pertained to commentary extracted from the Twitter platform. These were general comments searched for and collected at random and so represented general views about the users' water quality across the country. In this section, the search for comments was directed at WhatsApp users through a few questions related to water quality for the sources that they are using or have used (Table 3). The respondents (n = 100) were allowed to comment on different areas that they have visited or are visiting, thus one person could have comments, for example, for Johannesburg, Pretoria, Free State and Limpopo.

Responses to questions were received as text messages, which were then exported from the platform and converted into an Excel.xlx format document. This dataset was then loaded onto RStudio.

Table 3 Survey questions for WhatsApp

Provided below are some questions pertaining to water quality. Participants are required to give responses as truthfully and unambiguously as possible (try by all means to be descriptive in your responses).

When responding, please indicate your Province and District and number the responses using the same numbering system as below.

Questions:

1. Are you satisfied with the taste of drinking water in your area or is there any unpleasant taste to it, e.g. metallic, dirt-like, salty, etc?
2. Is the drinking water in your area clear, or does it contain some discolouration? If discoloured, what colour does it appear to be?
3. Does the drinking water in your area contain any suspended particulate matter, e.g. dust?
4. Do you trust the drinking water in your area?
5. Are you generally satisfied with the drinking water in your area?

3.2.1 Text data processing

The raw data was then processed such that it can be analysed by the algorithm. The data was firstly transformed into a corpus, and before any processing occurred, the mean text length of each of the responses was determined. Thereafter, the corpus was cleaned by removing stop words, numbers, punctuation, and white spaces and converting all words to lowercase. The corpus was surveyed for any spelling errors and appropriate corrections were made. For this study, a custom stopword dictionary was created to prevent the removal of negation words, which affected the interpretation of the data. The corpus was stemmed, where the terms were converted to their root words by removing the prefixes and suffixes, and vectorised.

The corpus was transformed into a Term Document Matrix (TDM). Such matrices are used in the bag-of-words model. Two observables were created by creating a data frame and placing terms and their respective frequencies alongside each other (Table 4). The data frame was subsequently organised in order of decreasing frequency and the top ten most frequent terms were chosen for further analysis.

Table 4 Word frequency for two observables

Word Frequency		
Water	Water	365
Not	Not	195
Satisfied	Satisfied	130
Taste	Taste	112
Clear	Clear	95
Trust	Trust	83
Yes	Yes	64
Tap	Tap	60
Sometimes	Sometimes	40
Sediment	Sediment	40
Drink	Drink	30

The following are examples of some of the codes used for text mining and analytics.

Sample of code for creating a term-document matrix:

```
#build a term matrix

tdm <- TermDocumentMatrix(corpus)
inspect(tdm)
tdm_m <- as.matrix(tdm)
```

Snippet of code for plotting a frequency term graph (based on Table 3.4) is:

```
#plotting
library(ggplot2)
library(ggthemes)
tdm_m$word <- factor(tdm_m$word, levels = unique(as.character(tdm_m$word)))
ggplot(tdm_m[1:20, ], aes(x = word, y = frequency)) + geom_bar(stat = “identity,” fill = “lightblue”)
```

The associations between the word “water” and other terms in the term document matrix were assessed as shown below. The code created a data frame consisting of factors for each of the terms and the association that correlates to the respective terms.

```
#word associations
associations <- findAssocs(tdm, “water,” 0.16)
associations <- as.data.frame(associations)
associations$terms <- row.names(associations)
associations$terms <- factor(associations$terms, levels = associations$terms)

#visualise associations
ggplot(associations, aes(y = terms))+geom_point(aes(x = water), data = associations, size=2) + theme_gdocs() + geom_text(aes(x = water, label = water), colour = “darkred,” hjust = -.15, size = 2) + theme(text =element_text(size = 6), axis.title.y = element_blank())
```

The other models created from various codes were for: word network, creating communities, plotting dendrograms and creating frequency word clouds.

3.2.2 Sentiment Analysis

To assess the sentiment in the data, the overall polarity in the data was first assessed. The data was scored for the sentiment in each of the responses obtained by calling on the polarity function of the Quantitative Discourse Analysis Package in R (qdap library) and a polarity distribution was plotted. A snippet of that code is as follows:

```
# Sentiment Analysis
library(qdap)
library(tm)
library(wordcloud)
library(ggplot2)
library(ggthemes)
options(stringsAsFactors = F)
sa <- read_excel(“C:\\Users\\uFeli\\OneDrive\\Documents\\cleaned water data.xlsx”)
sa.polarity <- polarity(sa$response)
```

```
ggplot(sa.polarity$all, aes(x = polarity, y =..density..)) + theme_gdocs() +
geom_histogram(binwidth = .25, fill = "lightblue," colour = "grey60," size = .2) +
geom_density(size = .75)
```

Subsequently, a comparison of the positive and negative words, based on the polarity scoring, was conducted. This was achieved using a comparison cloud. The syuzhet library was then used to analyse the sentiment in the text. This was achieved by first creating vectors from the data followed by a call to the get_nrc_sentiment function. This form of sentiment analysis was unsupervised and relied on the model to independently identify patterns. The overall trajectory of the opinions on the emotional valency scale was determined and an appropriate graph was plotted. To validate the sentiment data, the overall polarity of the data was analysed using the sentimentr package and a five-number summary (least value, lower quartile, mean, upper quartile and highest value) was obtained. A snippet of the code for this latter analysis is shown below:

```
library(sentiment)
sentiment <- sentiment_by(sa_df$response)
summary(sentiment$ave_sentiment)
```

Mean text length analysis

This analysis was conducted to determine the average length of the responses obtained from the respondents. The mean text length for each of the responses was found to be 53 characters. This value may be used as an indication of the willingness of respondents to provide their opinions regarding water quality.

Term document matrix generation

Sparsity in the term-document matrix was observed to be 99%. This value was high, indicating high dimensionality and possibly some attendant problems therewith. A high value was an indicator that the model might have overfitted the data or that there was high variance in the dataset.

A plot of associations between words in the corpus (not shown here), where the inclusive lower correlation limit was set to 0.16 showed that the that was strongly associated with the word "water" was "trust." The strength of connections amongst words were assessed using a word network (Fig. 4).

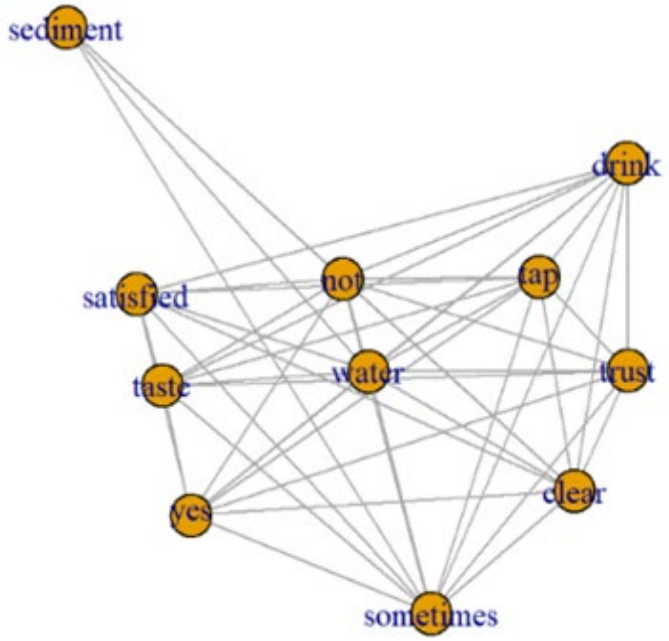

Figure 4 Word network showing connections amongst words

A cluster analysis of opinions was conducted and presented (Fig. 5). Clusters on the same heights have the same frequencies and those on different heights have significantly different frequency distances. From this plot, terms such as “water,” “not,” “satisfied” and “taste” would require further assessment since these clusters are in distinct positions on the plot relative to other clusters.

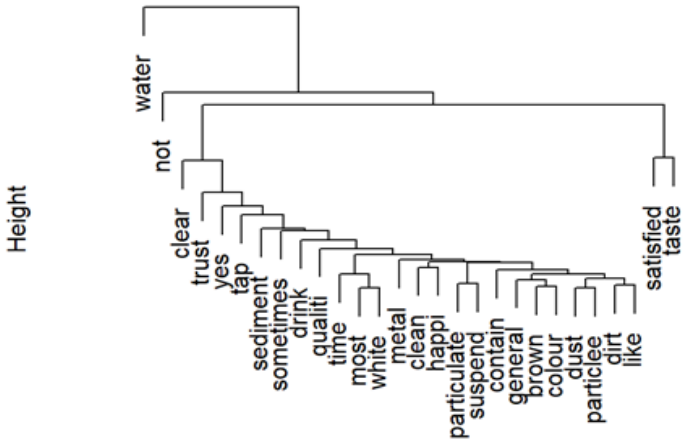

Figure 5 Cluster analysis of opinions

A word cloud plot was used to assess the commonalities existing amongst opinions shared regarding water quality (Fig. 6).

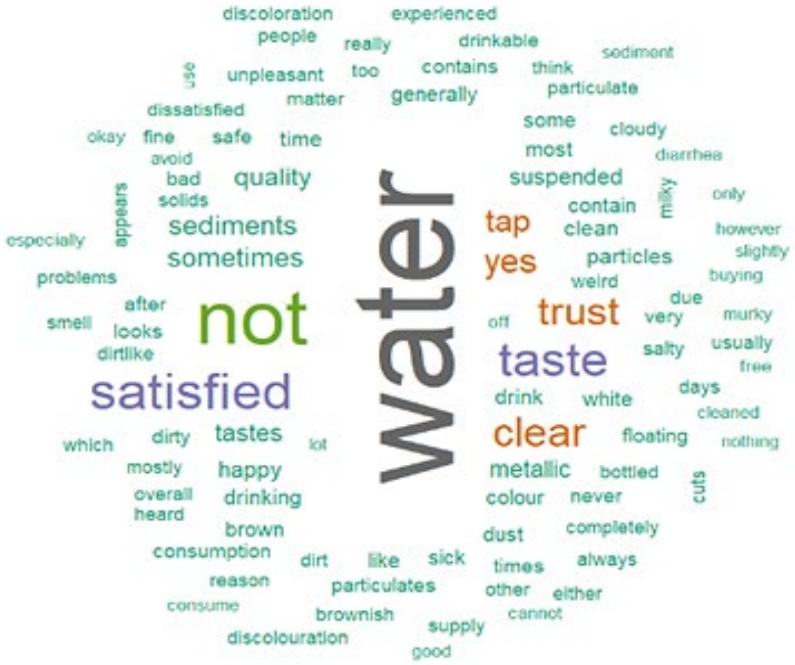

Figure 6 Word cloud indicating commonalities amongst opinions

For polarity distribution, the qdap library assigned the value -1 to negative sentiment, 0 to neutral sentiment, and +1 to positive sentiment and used valence shifters to assign the final polarity score. Polarity scores were centred around zero, and the mean score was found to be 0.0. The latter is expected since the plot is based on a standard normal distribution with $\mu = 0$. The density of the polarity scores appeared to be skewed to the right as most of the data were on the positive scale (not shown here).

Overall sentiment count was plotted (Fig. 7). The horizontal axis has the emotional valence for which the text was analysed. This plot shows that the overall sentiment was positive as the count for the positive sentiment is greater than the negative sentiment.

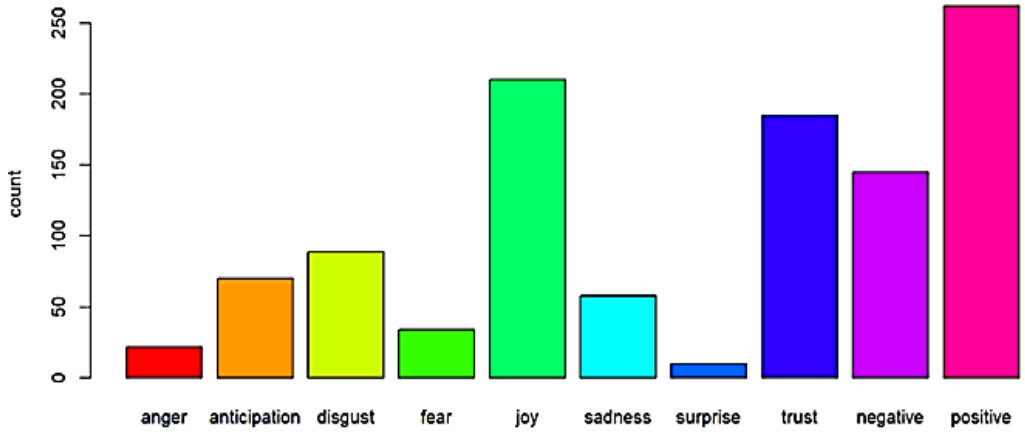

Figure 7 Sentiment scores for water quality

The above models were constructed and presented as demonstrations of the possibility of extracting text data from online platforms, namely WhatsApp (and Twitter before that), processing this data and modelling it based on machine learning techniques. Further demonstrations of this were presented in our previous work (Nuapia et al., 2021).

Having conducted text extraction and modelling, it is also important to assess the possibility of curating this data in an intermediary database. With continued growth in data available from different sources including online sources, data storage has grown in importance. The nature of properties of data in these sources presents interesting opportunities and challenges in the continual development of data storage capabilities. For instance, erstwhile, simple external drives were sufficient for storing data, but this is not sufficient in this age as volumes of data involved require larger and complicated repositories such as cloud facilities. This is presented in the following chapter in the context of textual data.

CHAPTER 4: CURATION IN INTERMEDIARY DATABASE

This chapter presents a two-pronged approach to data curation, namely: through coding and use of a third-party service. Text data is unstructured, making coding versatile in extracting it from online sources, modelling it and curating it in an intermediary database. This owes to the fact that coding gives more leverage to the user, allowing for some aspects to be created from scratch. A trial version of a pay-per-use third-party service provider, Fivetran, was also used to explore the possibility of using such a platform for data acquisition and storage. Capabilities and limitations of using either approach have been presented.

4.1 DATA STORAGE CURATION: USING CODING PLATFORM

Storage is one of the fundamental resources required in technological systems and software development. Data centres such as Google Cloud Storage, Amazon Web Services, Alibaba Cloud and Microsoft Azure have become a common feature and serve as the foundation of several different services (Fig. 8).

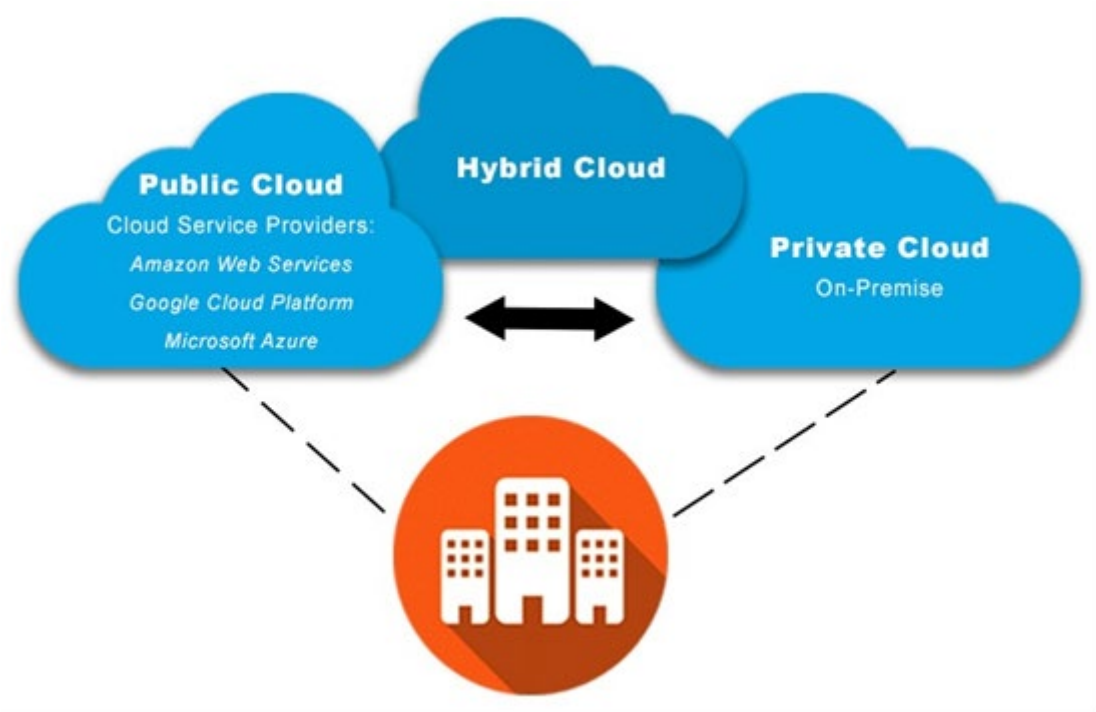

Figure 8 Cloud deployment models (source: AVI Networks)

Cloud storage is defined as a data deposit model in which digital information including documents, photos, music, videos and other forms of media are stored on virtual or cloud servers hosted by third parties. It allows for transfer of data to an off-site storage system and for access to them whenever they are needed (Rightscale State of the Cloud Report, 2019; CIO ThinkTank, 2022).

The choice of type of cloud storage to be used depends on the needs of the user, which includes number of users or accessibility considerations, security considerations (which also inform the quality control aspects) as well as cost considerations (Table 5).

Table 5 Advantages of different types of cloud storage

Private cloud storage	Public cloud storage	Hybrid cloud storage
Secure	Reliable	Secure
Scalable	Easily scalable	Scalable
Greater user control	Adequate monitoring	Greater user control
Expensive	Affordable	Affordable
Suitable for large enterprises	Suitable for individuals and mid-size companies	Suitable for individuals and mid-size companies

The data curation discussed here is based on Google Cloud which is one of the leading cloud providers and, with its supporting platform being the search engine and web browser, it is easily available and accessible. As it would be said about its platform, “Google is everywhere.” This ubiquity and availability gives Google Cloud the advantage to work well with programming platforms such as RStudio, structured query language (SQL), Python and customised or bespoke platforms (third party service providers). This has also been made easier by the ease of setting up email accounts, for instance, which, as will be demonstrated later, are required when accessing most cloud platforms.

4.1.1 Google Cloud Storage

Google Cloud provides three main services for different types of storage, namely: Persistent Disks for block storage, Filestore for network file storage, and Cloud Storage for object storage. These services are at the core of the platform and act as building blocks for the majority of the Google Cloud services and, by extension, to the systems that users build on top of it.

Block storage is the traditional storage type, both in the cloud and in on-premise systems. A Google Cloud Persistent Disk provides block storage and is used by all virtual machines in Google Cloud (Google Cloud Compute Engine). Persistent Disks can be viewed as analogous to USB drives in that they can be attached or detached from virtual machines and enable data persistence (promulgation) whenever virtual machines are started, stopped or terminated.

Filestore is the fully managed Google Cloud service that provides network file storage. Network file storage is similar in some respects to block storage, except that the disk storage is over the network.

Google Cloud Storage involves object storage which can be in different versions (object versioning) or fine grain permission (per object or bucket). Files can be added or obtained via a REST API and this can expand indefinitely with each object growing up to a huge memory scale. Different objects are grouped into unique “namespaces” called buckets and each bucket can hold multiple objects while a single object will belong to only one bucket. REST API stands for Representational State Transfer Application Programming Interface. An API is a product or program that sends information back and forth between a website or app and its user. A REST API generally uses HTTP to accomplish this mission in the background. While doing so, it breaks the request down into small packets of information that are sent back and forth, but does not keep or remember anything from the transactions thus making it safe and secure for users.

This model for storage is widely popular in cloud native systems due to its low cost, serverless approach and simplicity. The work of data replication, availability, integrity, capacity planning, etc. is then left to the cloud provider. The main drawback of object storage is that there is no other way to access the data besides the REST API.

The use of this type of storage for text data is demonstrated in the following sections, with the approach using the R platform to incorporate text mining, text data modelling, creation of objects (and buckets) and their storage or curation on Google Cloud.

4.1.2 Setting up GoogleCloudStorageR

GoogleCloudStorageR is an R library for interacting with the Google Storage JSON API. The JavaScript navigator object (JSON) is used for browser detection. It can be used to get browser information such as appName, appCodeName, userAgent, etc. The navigator object is the window property, so it can be accessed by window. For example, the following navigator object property will return the browser language:

```
let url = window.navigator.language;
```

Other examples of navigator object properties are presented below (Table 6).

Table 6 Some navigator object properties

Property	Description
appCodeName	Returns browser code name
appName	Returns browser name
geolocation	Returns a geolocation object for the user's location
cookieEnabled	Returns true if the browser cookies are enabled
onLine	Returns true if the browser is online
userAgent	Returns browser user-agent header

In order to run Google storage through R, a Google storage application (app) account is needed. It should be noted that Google Cloud Storage charges for storage, as do other platforms. A Project can be run in Google Project with a credit card added to create buckets, where the charges will apply.

Cloud Storage pricing is based on the following components:

- **Data storage:** the amount of data stored in buckets. Storage rates vary depending on the storage class of data and location of the buckets. Costs are also dependent on the length of time that large amounts of data will be kept.
- **Data processing:** the processing done by Cloud Storage, which includes operations charges, any applicable retrieval fees, and replication.
- **Network usage:** the amount of data read from or moved between buckets.

4.1.2.1 Authentication

To use any Google API, authentication, which involves concepts such as “service accounts,” “Oauth2.0,” and “scopes,” is required. The following steps were followed in this study to establish storage in Google Storage using R as a platform.

Step 1: Obtaining an email account – a Gmail account was set up. Any email account can be used. As indicated earlier, a Google account is a lot more versatile as a result of easy availability and accessibility.

Step 2: Activation of the Google Cloud Console
While logged in to the email account, Google Cloud Console was visited and terms of service agreed to followed by clicking on “Try for free” (Fig. 9).

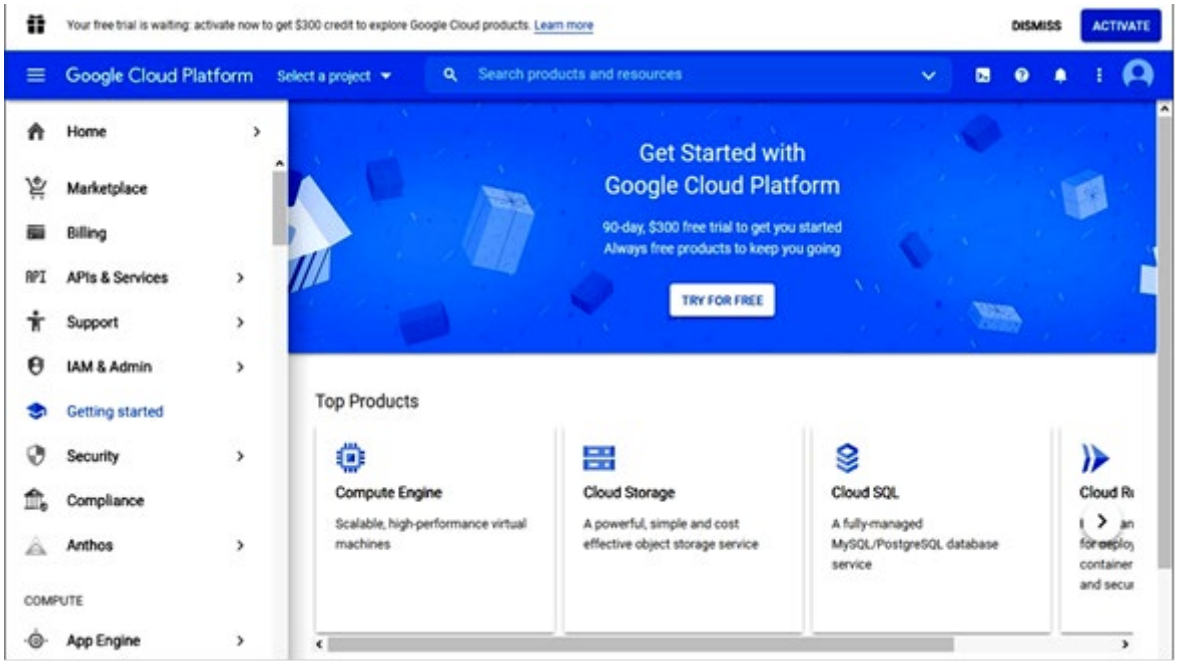

Figure 9 Google cloud platform setup

After that, the terms were accepted again, an address and a credit card information added. Because, the project has an *ID*, usually consisting of an adjective, a noun, and a number, this was stored in RStudio as vector:
my_project_id ← "<id>"

Step 3: Setting up the billing account

Return to the Google Cloud Console and look at the left column. Toward the top we clicked icon billing account (Fig. 10). A screen displaying "This project has no billing account" appeared. Click on a "link a billing account" and set the billing account to "My billing account."

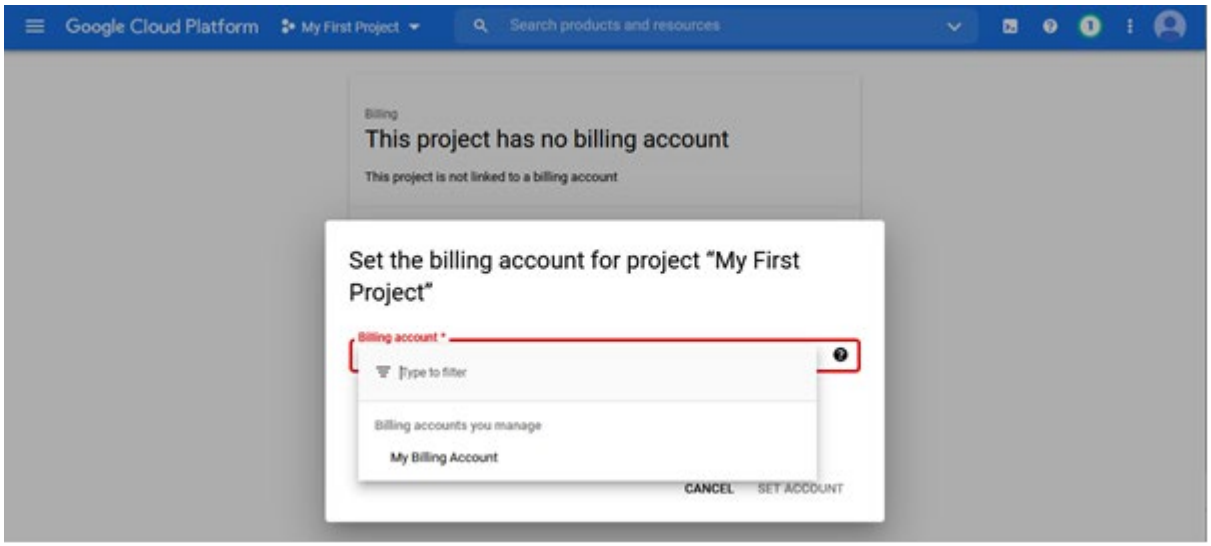

Figure 10 Creating billing account

Step 4: Activate additional APIs

To activate additional APIs, the following activities were carried out (Fig. 11):
Bring out the navigation menu on the left-hand side by clicking the little circle with the three horizonal lines in the top left of the screen. Then click on “APIs and services.” Scroll down to see list of the APIs which are enabled by default. These include the Google Storage API and a few others. Then open the navigation menu on the left again, and click on APIs and services. Then click on “Credentials” in the left pane.

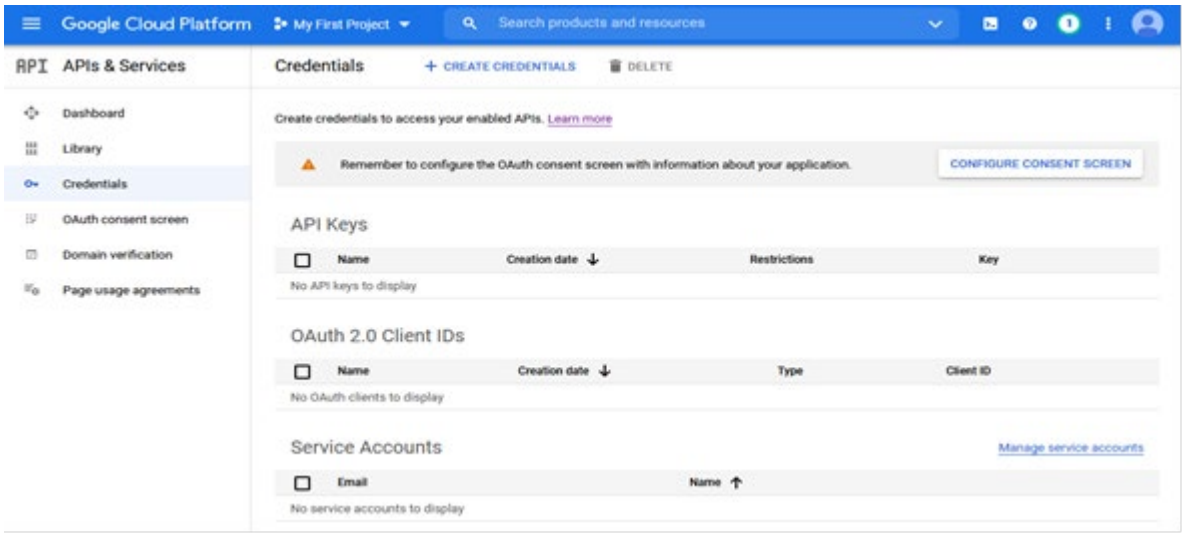

Figure 11 Activation of additional APIs

Step 5: Set up a service account

A service account was created by clicking on “+Create credentials” at the top and then choosing service account (Fig. 12). This was named “Water Research” and created by clicking “Create.”

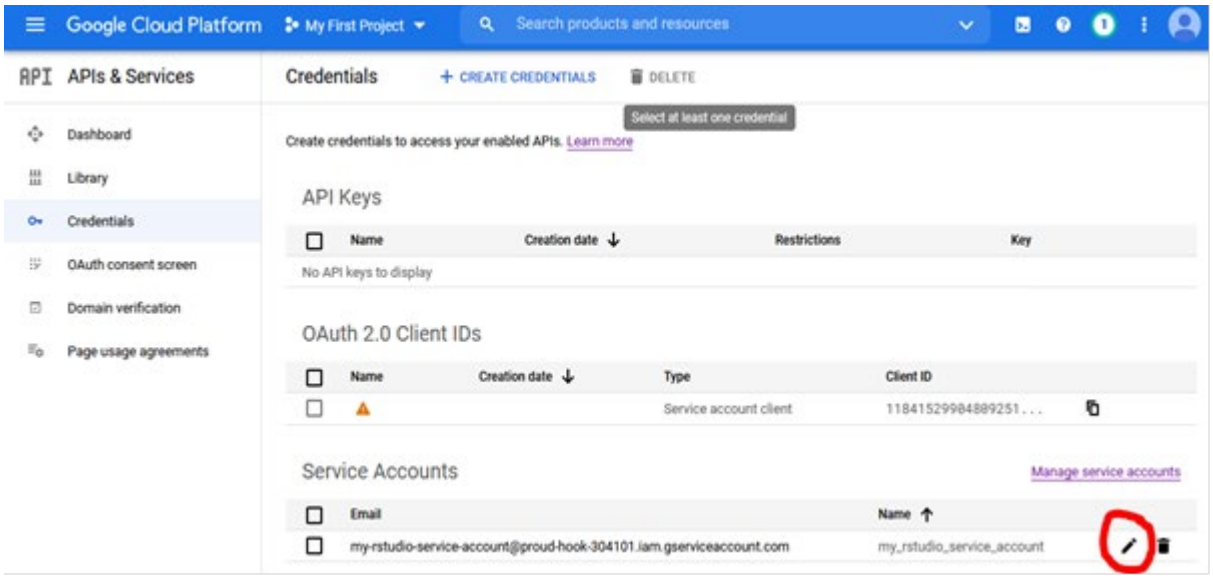

Figure 12 Setting up service account

Step 6: Download a json file with the service account key

A JavaScript Object Navigation (json) file containing the login details for the service account was then generated and saved as a file.

Step 7: RStudio Setting
The last step was to store the path to the json file in our .Renviron file so that RStudio can authenticate whenever we are working with GCS from R. This commenced by writing the following in the console:

usethis::edit_r_environ()

This opened a pane with the .Renviron file. At this stage, we added a line with the following: GCS_AUTH_FILE='<full path to the json file stored earlier>', ensuring that all the slashes in the filepath are forward slashes. The file was then saved, closed and RStudio restarted. The library googleCloudStorageR was loaded, auto-authenticated and was ready to communicate with the Google Storage account from within R. library(googleCloudStorage)

4.1.2.2 Creating and inspecting buckets

Google Storage is a file repository and it keeps files in "buckets." At least one bucket is needed to store files. To inspect the Storage account, the project id should be invoked (as will be shown later).

4.1.3 Working with GoogleCloudStorageR

GoogleCloudStorageR is a wrapper for the Google Storage API, that is it translates R input into uniform resource locators (URLs) that the API can understand. A uniform resource locator is a short string containing an address which refers to an object in the web. When the GoogleCloudStorageR functions are executed, they are actually sending GET and POST requests to Google and receiving responses in return.

As indicated earlier, to invoke the project id, the following procedure was followed.

```
my_project_id <- "<your project id>"
gcs_create_bucket("Water_Reasearch_2022," my_project_id, location = "SA")
gcs_list_buckets(my_project_id)
```

At this point, R can be instructed to recognise "Water Research 2022" as a default bucket. This has averted the case of adding the bucket id to every subsequent call.

```
gcs_global_bucket("Water_Research_2022")
```

More details about the bucket can be obtained with gcs_get_bucket()

```
gcs_get_bucket()
```

The bucket's file inventory is then obtained as follows:

```
gcs_list_objects()
write.csv(tweet, "tweet.csv")
gcs_upload("tweet.csv," name = "tweet.csv")
```

The contents are checked as follows:

```
gcs_list_objects()
```

Our bucket “Water Research 2022” in this case contained the file (or object) with comments from Twitter and a file of their codes and models. Separate files (or objects) can be added to this bucket as can separate buckets containing different files. This is dependent on the user’s preferences in relation to the project being conducted.

The Google Storage API handles only one file at a time, thus for bulk uploads a loop or an apply function is used. This was tested by downloading two random pdfs.

```
library(purrr)
download.file("https://cran.r-
project.org/web/packages/googleCloudStorageR/googleCloudStorageR.pdf,”
 "tweet.pdf")
download.file("https://cran.r-project.org/web/packages/tweet2/tweet2.pdf,”
 "tweet2.pdf")
my_pdfs ← list.files(pattern = "*.pdf")
map(my_pdfs, function(x) gcs_upload(x, name = x))
```

4.1.3.1 Downloading files

Downloads are performed with gcs_get_object(). It was saved under a different name, using the parameter saveToDisk.

```
gcs_get_object("tweet.csv,” saveToDisk = "tweet.csv")
```

To download multiple files, a loop or map is used. For instance, if all pdfs in the bucket are to be downloaded:

```
contents ← gcs_list_objects()
pdfs_to_download ← grep("*.pdf,” contents$name, value = TRUE)
map(pdfs_to_download, function(x) gcs_get_object(x, saveToDisk = x, overwrite = TRUE))
```

4.1.3.2 Deleting files

The files in the bucket can be deleted with gcs_delete_object():

```
gcs_delete_object("tweet§.csv")
```

To delete several files, a loop or map is required. For example:

```
contents <- gcs_list_objects()
map(contents$name, gcs_delete_object)
```

4.1.3.3 Dealing with folders

It is not possible to have folders in Google Storage. Rather, files in a bucket are stored side by side in a flat structure. It is possible to imitate a folder structure by adding prefixes to filenames (using forward slashes). This is important in keeping files organised and for instances where a subset of files in a bucket have to be processed. This is illustrated as follows by creating two folders in the working directory. The one folder is for csv files and the other for pdfs.

```
dir.create("csvs")
dir.create("pdfs")
csv_files <- list.files(pattern = "*.csv")
pdf_files <- list.files(pattern = "*.pdf")
file.copy(csv_files, ."/csvs")
file.copy(pdf_files, ."/pdfs")
```

To upload the folders, vectors of the files were made in each folder, with the addition of "full.names=TRUE to list.files()" to include the folder name.

```
csvs_to_upload <- list.files(."/csvs," full.names = TRUE)
pdfs_to_upload <- list.files(."/pdfs," full.names = TRUE)
```

The slash part "/" of the file path was then removed before saving to Google Storage. To do this, "gsub()" was used. A regular expression (i.e. regex or regexp) is a sequence of characters that specifies a search pattern in text (Seewald, 2020). Usually, these are used by string-searching algorithms such as those for "find" or "find and replace" operations on strings, or for input validation. Since .and / are special characters in regex, a backslash was used for them.

```
map(csvs_to_upload, function(x) gcs_upload(x, name = gsub("\\.\\/," "," x)))
map(pdfs_to_upload, function(x) gcs_upload(x, name = gsub("\\.\\/," "," x)))
```

The bucket contents can then be checked to ascertain if the files are in "folders."

```
gcs_list_objects()
```

The contents of one bucket "folder" can be downloaded as follows:

```
contents ← gcs_list_objects()
folder_to_download ← grep("csvs/*," contents$name, value = TRUE)
map(folder_to_download, function(x) gcs_get_object(x, saveToDisk = x, overwrite = TRUE))
```

If a "csvs" folder does not exist, R would return an error message. This is because the function "gcs_get_object() function cannot create new folders on the hard drive. Thus, if there is a tree of subfolders in the bucket this can be maintained (the tree format, that is) by having a destination folder with the same folder when downloading. To circumvent this, the forward slash can be replaced by something such as an underscore and then reconstructing the folder structure at a later stage.

```
contents ← gcs_list_objects()
map(contents$name, function(x) gcs_get_object(x, saveToDisk = gsub("/," "_," x), overwrite =
TRUE))
```

To upload a folder that contains many subfolders, the following steps can be taken in which two subfolders are created in the csvs folder and some files put in there.

```
dir.create(."/csvs/folder1")
dir.create(."/csvs/folder2")
write.csv(WaterQuality1, ."/csvs/folder1/waterquality1.csv")
write.csv(WaterQuality2, ."/csvs/folder2/waterquality2.csv")
```

To upload the csv folder with subfolders the parameter “recursive = TRUE” is added to the list.files() call. The filepath is cleaned as illustrated previously.

```
files_and_folders ← list.files(.”/csvs,” full.names = TRUE, recursive = TRUE)
map(files_and_folders, function(x) gcs_upload(x, name = gsub("\\.\\/,” ",” x)))
```

The bucket can be inspected again using:

```
gcs_list_objects()
```

The preceding discussions have presented an approach of using Google Storage (based on R) for secure and permanent curation of data and models generated from various sources, including images, videos, audios and text mining data. As indicated previously, most platforms such as Twitter offer limited timeframes to access tweets e.g. only up to 10 days. This means that the data and model files should be updated continuously. The approach presented here for data curation has the advantage of permanently storing data and models compared to other online approaches that provide capacity for conducting modelling (on a shared basis), but only store the files temporarily.

Data extraction from sources can be done in real time using third-party service providers such as Fivetran. However, this is limited only to data extraction and curation in the preferred destination warehouse after which the data user has to conduct the modelling. Essentially, the third-party plays the role of a data shuttle. This option is discussed in the following section.

4.2 DATA STORAGE CURATION: USING THIRD-PARTY SERVICE

As indicated above, the possibility of online data extraction was explored using a trial version of a third-party service, Fivetran (https://www.generalcatalyst.com/gcamplified/fivetran-making-data-flow/ Accessed on 15 May 2022).

4.2.1 Architecture

Generally, in Fivetran the building blocks of data organisation are tables and schemas (the equivalent of buckets in the previous section). While a table is a file organised by rows and columns, a schema is a folder that contains multiple tables. In Fivetran, connectors are used that create and manage their own schema individually. A connector reaches out to the data source (e.g. Twitter in this study), extracts data from it, and writes it to the preferred destination warehouse (Fig. 13). There are two types of connectors, namely a push connector and a pull connector.

Pull connectors actively retrieve, or pull, data from a source. Fivetran connects to and downloads data from a source system at a fixed frequency. It uses an SSL-encrypted connection to the source system to retrieve data using a variety of methods including web service APIs via REST, depending on the source system. SSL stands for Secure Sockets Layer and it is the standard technology for keeping an internet connection secure and safeguarding any sensitive data that is being sent between two systems, preventing criminals and unauthorised people from reading and modifying any information transferred, including potential personal details.

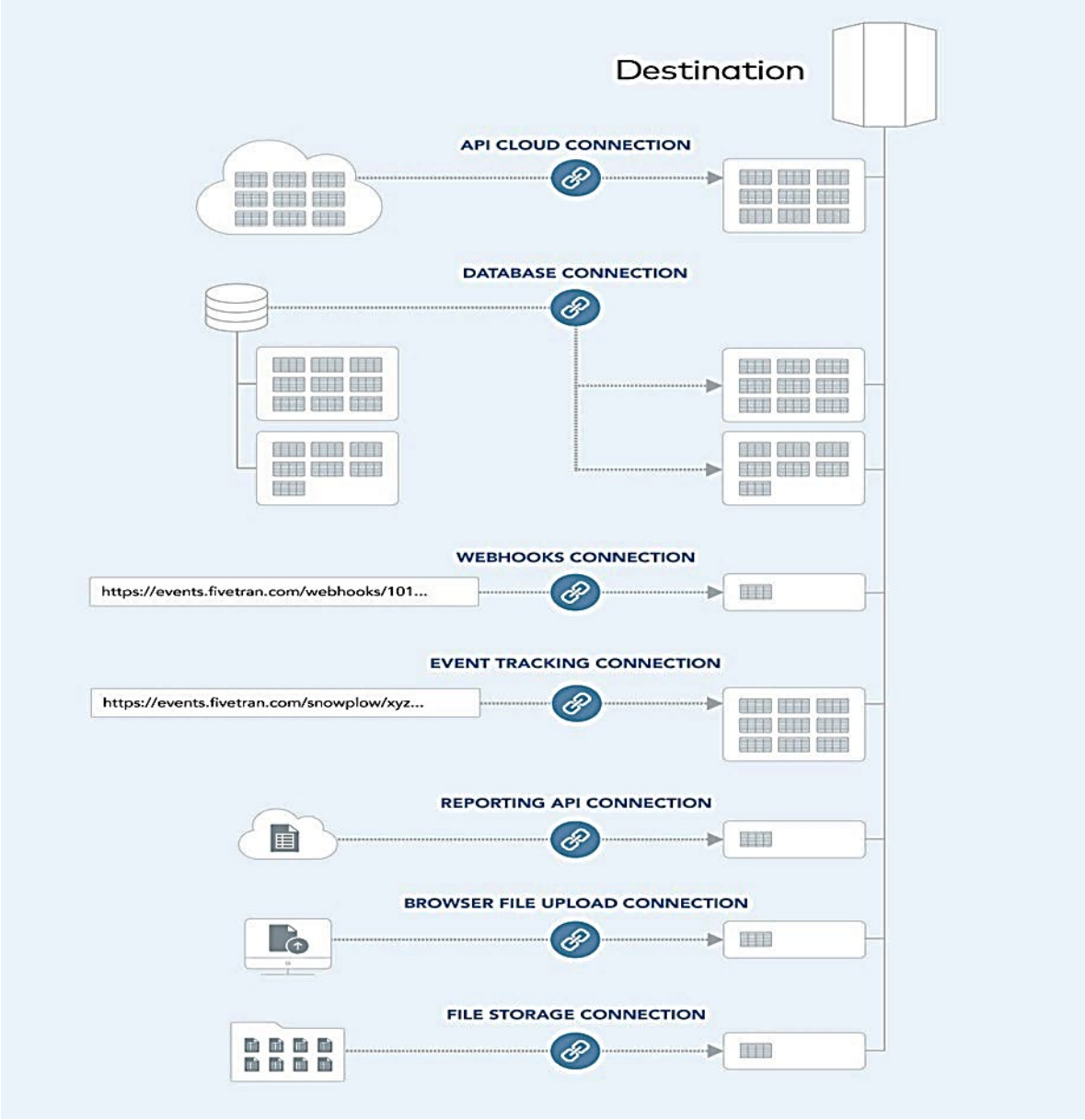

Figure 13 Different connectors for each of Fivetran's supported connector types
(https://www.generalcatalyst.com/gcamplified/fivetran-making-data-flow/ Accessed on 15 May 2022)

In push connectors, such as Webhooks and Snowplow (Fig. 13), source systems send data to Fivetran as events. The push connector pipeline is as follows:

1. When the events are received in the collection service, they are buffered in the queue.
2. The event data are stored as JSON in cloud storage buckets.
3. During the sync, the data is pushed to the user's destination.

The former case of push connectors best suits the situation related to this study as data is drawn from the source (e.g. Twitter and WhatsApp in this instance) rather than processed and sent by the source (i.e. the push case) as this is not possible.

The architecture of a platform such as Fivetran is that it extracts data from the source and loads it as raw data in the destination warehouse. The transformation of the data, e.g. modelling is left to the user to perform post loading. This approach is commonly called the extract-load-transform (ELT) pipeline.

For their part, Fivetran intends to deliver the information in a cleaned and normalised schema, called the canonical schema, at the lowest level of aggregation. Erstwhile, pipelines would pre-aggregate the data and scale it down to what would be considered a workable size. This has been changed to allow for more control on the part of the user.

Fivetran regularly maintains the connectors and evolve the canonical schema in accordance with changes occurring in the source systems. In the case of Twitter, this is a continuous data extraction extending beyond the standard 10 days of availability of data on that platform. This in turn means a continuous update of the canonical schema to which the data is loaded in the Google Cloud. Then, it is left to the user to continuously update the transformation and modelling of new data as it is received.

The steps followed are simply:

- Create one or more source connectors (in our case, Twitter).
- Connect target destination (in our case, Google Cloud).
- Fivetran then conducts the extraction, transformation, and loading of data to the destination.

Our approach in this study to use a third-party service provider such as Fivetran is predicated on the pathways shown in solid blue lines while that based on independently accessing data from the source, processing it and storing it on the cloud is shown by the broken red lines (Fig. 14). The steps and procedures to conduct the latter have been described in the preceding section where the RStudio platform was used.

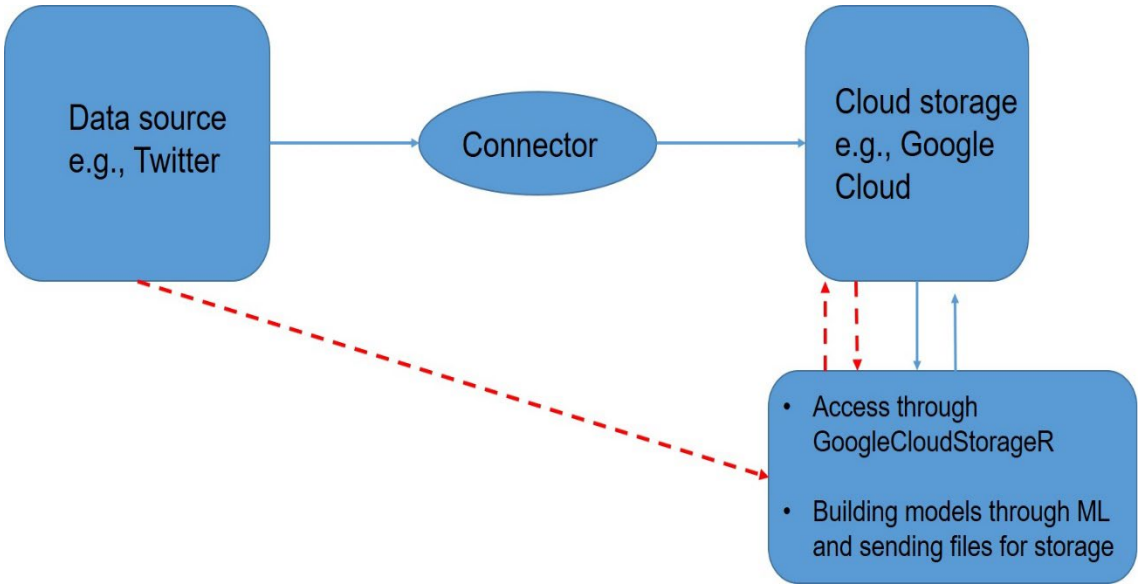

Figure 14 Data attainment curation using third-party such as Fivetran (solid blue lines) and independently using RStudio (broken red lines)

4.2.2 Data process management

Destination schema are usually named after the connector name or the connector that created them. The destination name is permanent and cannot be changed. Fivetran creates, delivers and manages the base tables to the user's destination. Base tables are direct replicas of tables from the source that are used as building blocks for the analysis conducted in the destination.

Updates are conducted on the internal representation to identify any schema changes in the source. The internal representation is built on Fivetran's platform, external to the user's destination and as such

any changes can only be done outside the destination. Data loading to the destination occurs using queries, e.g. those based on the structured query language (SQL). This is a commonly used language for databases and database management (Holywell, 2022). Fivetran uses queries to load data into the user's destination. Since service providers (e.g. Google in our case) charge for storage, they will charge when Fivetran loads data into the destination. The costs are dependent on two main factors, namely:

- *Data volume*: costs are a function of the amount of data, thus loading large amounts of data will cost more.
- *Sync frequency*: syncing data frequently may cost more, depending on how often updates happen in the source. Syncs can be done at intervals of minutes, hours or days.

For other types of data, the creation of base tables is quite easily achievable. A base table is a physical structure that contains stored records. This structure can have any physical format as long as it can be presented to the user as a collection of rows and columns. Thus, this is easily done for structured data such as analytical data or other similarly structured data such as financial records, human resources data and marketing data. This type of data works well with SQL as queries can be created using table headings and the data pertaining to them called. For example, Fivetran would be able to fetch base tables such as Table 7 from an online source, e.g. a cloud-based shared site for a company. Using SQL, each variable such as "Expenses" can be assessed for individual selected plants and with respect to plant leaders. As such the queries are structured around the table variables.

Table 7 Sample of structured data format

Plant no.	Variable 1	Variable 2	Variable 3	Expenses (ZAR)	Lead Scientist

In our study, the data is not in the above format, but is rather in the form of comments (Table 8). This means that there is not even a base table to work from or to support table abstraction, making it difficult for third party service providers to extract such data using languages such as SQL.

Table 8 Sample of unstructured text data (online commentary Twitter, WhatsApp)

Comment #1 The water from my tap is brown
Comment #2 We're currently using water from a river
……
……
……
Comment #200 I've been drinking bottled water since two years back

It is worth noting here that there are several options for accessing data from sources and storing it in preferred data warehouses or data lakes using third party services, notwithstanding the above challenges related to unstructured text data. With the increase in data generated by various sources, there has also been a correlated growth in companies that provide such third-party services using the cloud.

The cloud is inelastic, allowing for gathering large amounts of data from a multitude of data sources and scaling to support a large number of users and workloads. The advent of most of these companies has seen a growth in listings on stock exchanges such as the Nasdaq, forming a number of what are referred to as special purpose acquisition companies (or SPACs) which in itself is a new concept in company

listings. In some cases, the new companies become targets for takeover and purchase by bigger established technology companies.

The growth trend in such companies is likely to continue for the foreseeable future, as large volumes of data are being generated continuously that will require access, manipulation and storage. As this occurs, so do concerns increase around data quality, integrity and security.

CHAPTER 5: QUALITY CONTROL

This chapter presents an overview of quality control aspects with respect to storage and maintenance of its integrity when using cloud storage. As any other component of computing, it is important to protect systems and, subsequently, data from being compromised. This will ensure data quality, data integrity and data quality. There is a shared responsibility here between the cloud providers and users, with the former protecting the underlying cloud infrastructure while the latter protect data and cloud-deployed assets.

5.1 DATA QUALITY INTEGRITY

A cloud data warehouse is an internet-based repository of data derived from diverse sources. These sources include internal organisational operational systems (e.g. data generated from research work) and external (e.g. data drawn from online resources). In water quality research, such as that presented pertaining to the study, the former sources would typically be analytical data collected from the laboratory, while the latter source would be text data collected from online platforms such as Twitter and WhatsApp.

Data quality control is easier for internally sourced data as the user has more control on processes involved such as sampling, analysis and data analysis. Externally sourced data poses a quality challenge owing to the lack of control by the user. This is more pronounced in text type of data such as that obtained from interviews and opinion polls. Some websites tend to filter comments coming through to them, presenting a risk of having comments that did not originally carry the meanings inferred from them at the time of collection. This may be a necessary practice from the part of the sources as they tend to contend with prejudice, hate speech and other communication challenges.

In our previous study on citizen science (Nuapia et al., 2021), it was indicated that during the interviews conducted, some of the respondents expressed concern at the potential political backlash for any negative comments that they could give with respect to their water resources. Thus, opinions collected from such individuals would contain some bias and may not reflect the true state of their water resources.

Social media is also quite prone to views and opinions of influencers, who usually have a large following on those platforms. A typical example here would be that of Tesla founder, Elon Musk who tweeted in 2021 about “a dog reaching for the moon” which was a veiled reference to speculation on the prospects of the cryptocurrency Dogecoin. With a following of over 80 million on Twitter, his tweet led to “crowding” on Dogecoin, raising its price by tenfold. This was by no means reflective of the true underlying performance of the cryptocurrency. This, together with other misdemeanours such as fake accounts and commentary, poses a challenge when dealing with the quality and integrity of online data.

In this study, data quality and integrity could not be ascertained comprehensively as the comments were for disparate situations and quite general. While this aspect was important, the main focus remained the ability to demonstrate that text mining can be used to obtain data from online platforms such as Twitter.

5.2 DATA SECURITY

One of the foremost challenges in computing is cybersecurity. Established systems usually tend to be safer as a result of experience, research and constant efforts towards protection. However, the current

dispensation in computing has seen many changes and innovation within a short space of time with most of these occurring in the past 15 years or so. Large volumes of data have been produced and innovative methods of handling and managing such data are under constant development. Among these is cloud data warehousing. As much as the service providers would try to provide a secure environment, hackers have also become complicated in their approach, making data that is stored in such systems to be under potential threat of being compromised.

There a several instances where databases have been hacked and data compromised. Perhaps the most prominent recent case is that of the Colonial Pipeline Company that supplies fuel in the US. It suffered a ransomware cyberattack in May 2021 that impacted computerised equipment managing the pipeline, causing severe shortages of fuel in the south eastern US. In early 2022, Dischem SA also suffered a hacking attack on its customer database, compromising millions of customers' personal information. A typical example of newer and innovative computing tools coming under security threat is the hacking of several cryptocurrency wallets in the US in 2021, resulting in the loss of millions of dollars in crypto coins. This is concerning especially after most of the companies involved have always touted the blockchain network (on which crypto transactions are conducted) as very safe and not prone to hacking. In fact, ransomware operates on the blockchain. There has also been a case, in May 2022, of hacking of the latest technology, the Metaverse, resulting in tokens (accounts) being stolen.

It is worth noting here that blockchain technology and the Metaverse (based on both virtual and augmented reality) will be the main technologies in the near future. While the former is strongly associated with crypto assets, its networking concept of decentralisation is gaining preference over the conventional centralised control system. It can be observed from here that in future, databases are likely to adopt this approach resulting in fast processing and cost reduction related to stages in data handling.

In relation to this study, the storage of the text files and modelled data in Google Cloud is backed by storage in conventional systems such as hard drives and portable drives. This is possible as a result of the small size of the data. As the data size increases, storage will be more reliant on the cloud storage platforms and challenges of security will be higher. It would also be worth considering the use of a cloud access security broker (CASB) software. The software is placed between cloud service users and cloud applications, monitoring cloud usage and ensuring security in the process. This looks to have a potential positive growth outlook in this dispensation of increased cloud usage.

CHAPTER 6: CONCLUSION RECOMMENDATIONS

This chapter presents conclusions drawn from the study, encompassing aims and findings. It also provides some implications that can be drawn from the study, both theoretical and practical, and lastly some recommendations that will be useful for benchmarking future studies and decision-making.

6.1 CONCLUSION

The study explored how text mining of online data from platforms such as Twitter and WhatsApp could enhance hydroinformatics, particularly water quality. It explored the chain of custody, starting from extracting text data to processing and modelling this data with respect to opinion and sentiment analysis of water quality and its curation in an intermediary database on a cloud-based platform.

Text searching, extraction and processing produced term document matrices, which were further simplified to convert unstructured text data into vectorised structured formats, were then modelled using machine learning tools. The resulting models of opinions and sentiments revealed the general views that users and respondents had regarding the quality of their water. Positive, negative and neutral opinions and sentiments were drawn from the comments. These were further analysed to reveal connections, for instance, trust or distrust of their water providers, satisfaction or dissatisfaction. Word clouds and cluster analysis were also used to enhance delineation of opinions and sentiments. This way, it was possible to deduce the positivity, negativity or neutrality thereof. These findings would imply that it is possible to extract and model text data from various other online platforms such as emails, blogs and chat platforms.

Two approaches were considered for the curation of data in an intermediary database, namely using a coding platform (based on RStudio) and a customised platform or third-party service. The former approach was based on data obtained through Google Cloud storage accessed from the RStudio platform. Twitter and WhatsApp comments were stored in files created on the cloud prior to modelling and curating the resulting models in the same cloud warehouse. The customised platform was based on Fivetran (a number of other providers can also be used) through a limited trial version. It was demonstrated that it is possible to extract data from an online source and store it on the cloud warehouse, e.g. Google Cloud in this case. The data processing from there remains the responsibility of the user. However, the format of text data that could be extracted was limited to base tables rather than random comments as would be obtained from Twitter and WhatsApp. For base tables, the structured query language (SQL) is commonly used, abstracting data using query commands referring to table labels and variables. Such queries cannot be made for general and unformatted text such as that on Twitter and WhatsApp.

In terms of flexibility, the use of coding was found to offer more flexibility, as the user can control changes. For instance, the formats for data tables can be manipulated using code at creation and during storage. Search formats can also be changed and optimised by use of code, which is not easily achievable from customised software such as Fivetran as the back-end code is fixed and not accessible or editable.

Aspects of data quality control, namely data quality, data security and data integrity were presented. These were discussed with respect to their potential to impact the curation of the data under consideration. As a developing technology, cloud storage faces a number of potential threats with respect to data quality control aspects. It could also be observed that there is lack of standardisation in the technology, with different service providers offering different charges for storage and operational support that could be available. Thus, drawing up a due diligence checklist for cloud computing,

covering sub-categories such as terms of service, charges, security, and data backup can go a long way towards making sure that important aspects are addressed when researching a cloud data warehouse provider.

6.2 RECOMMENDATIONS

Real time collection of data

As the findings have indicated, coding gives far more leverage in conducting text mining from online platforms and the development of an intermediary database for storing the data within the entire chain of custody. This gives the user more control over the whole process compared to using customised or third-party platforms. Providing effective linkages between text mining; processing of the text data; conducting analytics on the text data to develop useful models such as those based on opinions and sentiments; and storing the data on cloud platforms remains an important aspect to which future studies should be directed. This will make it possible to access text data from online platforms on a continuous rather than batch basis (Fig. 15).

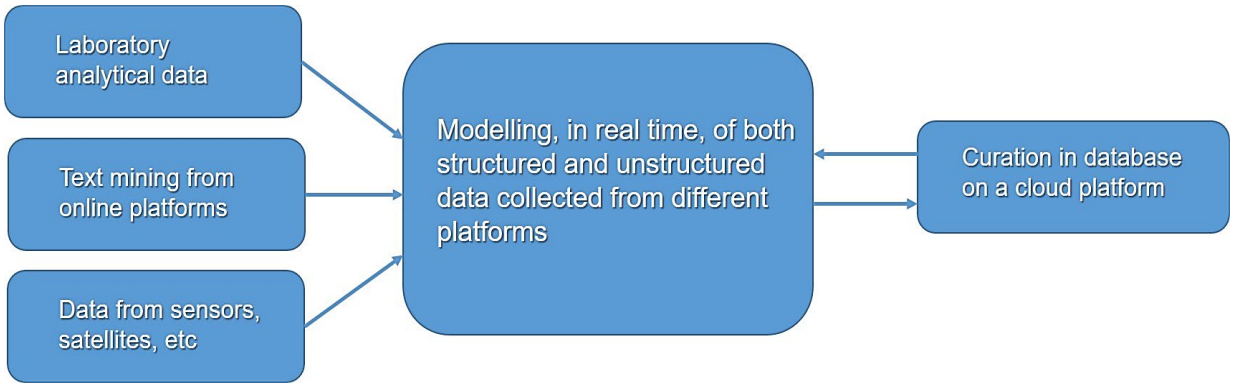

Figure 15 Layout of recommended automated approach to real time collection of text and other data, modelling and curating it on cloud platform

Text mining of indigenous languages

This is important to consider, especially in instances where a community-based case study is used. For instance, a WhatsApp group can be set up and respondents requested to answer in their own language, e.g. Setswana, Sesotho, Afrikaans, etc. This can then be followed by text mining and modelling developed around the language. This will yield more appropriate outcomes with respect to opinions and sentiments as respondents will likely be more comfortable in their responses. It is also possible to conduct text mining of different indigenous languages simultaneously and drawing comparisons in their expressions. This would be useful in communities where people speak different languages. Further, consideration of informal and yet popular languages as Tsotsitaal and Iscamtho should also be made.

Broadening the scope of hydroinformatics

Owing to time constraints, the study focused on one aspect of hydroinformatics, namely water quality of drinking water. Other aspects can also be pursued in future studies and these include: flooding events, droughts, weather elements (e.g. humidity and temperature), climate change, agricultural indicators (e.g. changes in crop yield and nutrient availability), and pollution among others. Because of their varied nature, these aspects will require a longer time frame on the project and ideally a specific study area. For instance, collating commentary on all these aspects over a period of a year or so in KwaZulu Natal where flooding events have increased lately would yield useful insights. An area of a few villages can be used as a case study, providing several respondents who will be able to give

commentary over a protracted period of time. Those comments can then be processed in real time (as suggested above), enabling patterns to be identified that will be important as early warning signs once there is a sufficient build-up of them. Other citizen data such as videos and images can also be incorporated to extend the data beyond just text.

REFERENCES

ABUHAY T M, NIGATIE YG, KOVALCHUK SV (2018). Towards predicting trend of scientific research topics using topic modeling. Procedia Computer Science **136** 304–310.

ALGHAMDI R, ALFALQI K (2015) A survey of topic modeling in text mining. International Journal of Advanced Computer Science and Applications (IJACSA) 6(1).

BROWNLEE J (2019) Probability for machine learning. Mastery Machine Learning Series (ebook).

BRUCE C, METZLER D and TREVOR S (2009) Search engines: Information retrieval in practice. Addison-Wesley, Boston, MA. 136pp.

COSKUN M, OZTURAN M (2018) #europehappinessmap: A Framework for Multi-Lingual Sentiment Analysis via social media Big Data (A Twitter Case Study). *Information **9*** 102.

GAMAL D, ALFONSE M, M EL-HORBATY ES, M SALEM AB (2019) Analysis of Machine Learning Algorithms for Opinion Mining in Different Domains. *Mach. Learn. Knowl. Extr 1* 224–234.

GENC-NAYEBI N, ABRAN AA (2017) systematic literature review: Opinion mining studies from mobile app store user reviews. *J. Syst. Softw. **125*** 207–219.

HORTO A, STAAB S and STUMME G (2003) Text clustering based on background knowledge. Institute of Applied Informatics and Formal Descriptive Methods, University of Karlsruhe, Germany. 35pp.

LOCHBAUM KE, GROSZ BJ and SIDNER CL (2000) Discourse structure and intention recognition. In Dale, R., Moisl, H., & Somers, H. (Eds.), Handbook of Natural Language Processing, Marcel Dekker, Inc., New York. 123-146 pp.

JELODAR H, WANG Y, YUAN C, FENG X, JIANG X, LI Y, ZHAO L. (2019). Latent Dirichlet allocation (LDA) and topic modeling: Models, applications, a survey. Multimedia Tools and Applications **78**(11) 15169–15211.

JU S, LI S, SU Y, ZHOU G, HONG Y, LI X (2012). Dual word and document seed selection for semi-supervised sentiment classification. In Proceedings of the 21st ACM International Conference on Information and Knowledge Management, Maui, HI, USA 2295–2298.

MANNING, CHRISTOPHER D, PRABHAKAR R and HINRICH S (2008) Introduction to information retrieval, Cambridge, University Press, MA. 96pp.

NISBET R, ELDER J and MINER G (2009) Handbook of statistical analysis and data mining Applications. Elsevier, Burlington, MA. 123pp.

NUAPIA Y, NDHLOVU L, NGUNDU L, KHOZA P, ETALE A, CUKROWSKA E, TUTU H (2021). Imagining Solutions for Extracting Further Value from Existing Datasets on Surface and Groundwater Resources in Southern Africa. WRC Report No. TT 842/20. Available from WRC Knowledge Hub: www.wrc.org.za

POZZI FA, FERSINI E, MESSINA E, LIU, B (2016) Beyond Sentiment: How Social Network Analytics Can Enhance Opinion Mining and Sentiment Analysis. In *Sentiment Analysis in Social Networks*, 1st ed.; Morgan Kaufmann Publishers Inc.: Los Angeles, CA, USA. 134pp.

RATINOV L and ROTH D (2009) Design challenges and misconceptions in named entity recognition. In: Proceedings of the Annual Conference on Computational Natural Language Learning (CoNLL). 147-155 June 2009, Colorado.

RENDERS J (2004) Kernel methods for natural language. In: Processing Learning Methods for Text Understanding and Mining, 26-29 January 2004, Grenoble, France.

SMILEY D and PUGH E (2009) Solr 1.4: Enterprise Search Server. Packt Publishing, Birmingham, England, UK. 65pp.

SMITH A and HUMPHREYS M (2006) Evaluation of unsupervised semantic mapping of natural language with Leximancer concept mapping. Behavior Research Methods, **38**(2) 262.

SOUMEN C (2002) Mining the web: Analysis of hypertext and semi-structured data. Morgan Kaufmann, San Francisco. 56pp.

VIDHYA KA and AGHILA G (2010) Text mining process, techniques and tools: an Overview. International Journal of Information Technology and Knowledge Management **2**(2), 613-622.

POZZI FA, FERSINI E, MESSINA E, LIU, B (2016) Beyond Sentiment: How Social Network Analytics Can Enhance Opinion Mining and Sentiment Analysis. In *Sentiment Analysis in Social Networks*, 1st ed.; Morgan Kaufmann Publishers Inc.: Los Angeles, CA, USA. 134pp.

APPENDIX 1: OPINION SENTIMENT MODELS

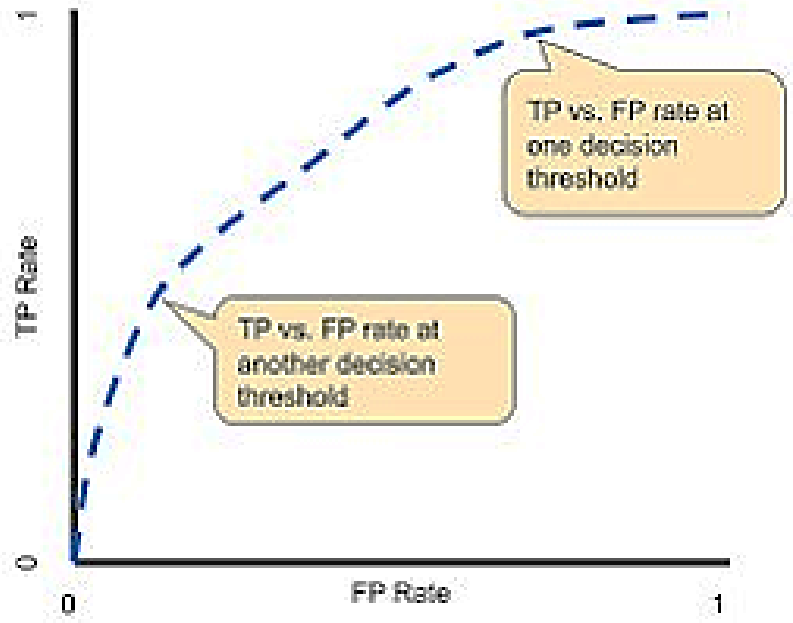

Figure 16 Sample of receiver operating curve (ROC) showing true positive (TP) vs. false positive (FP) rate at different classification thresholds

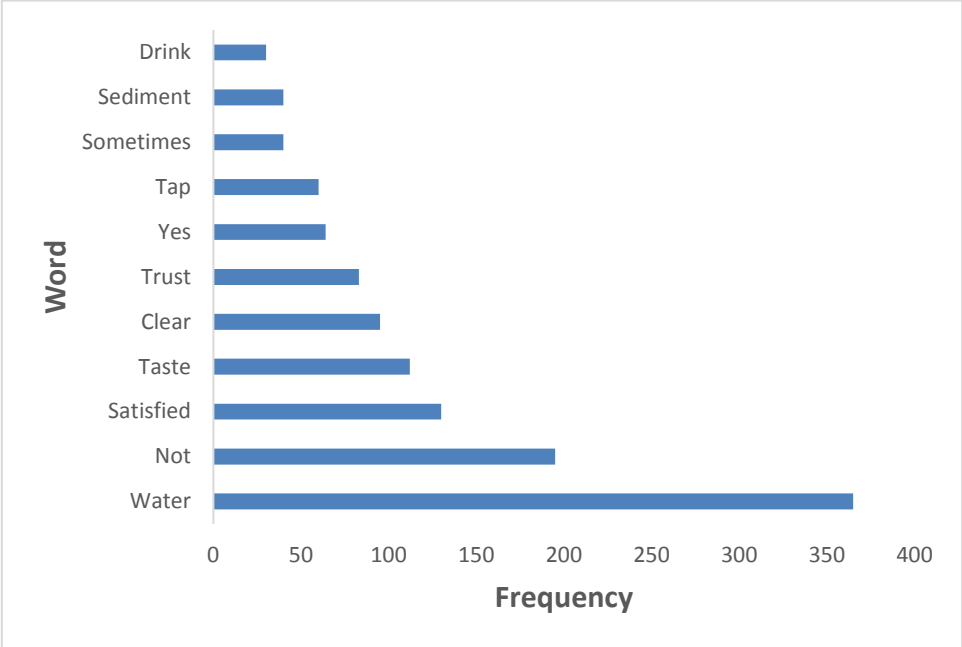

Figure 17 Frequency of words plot

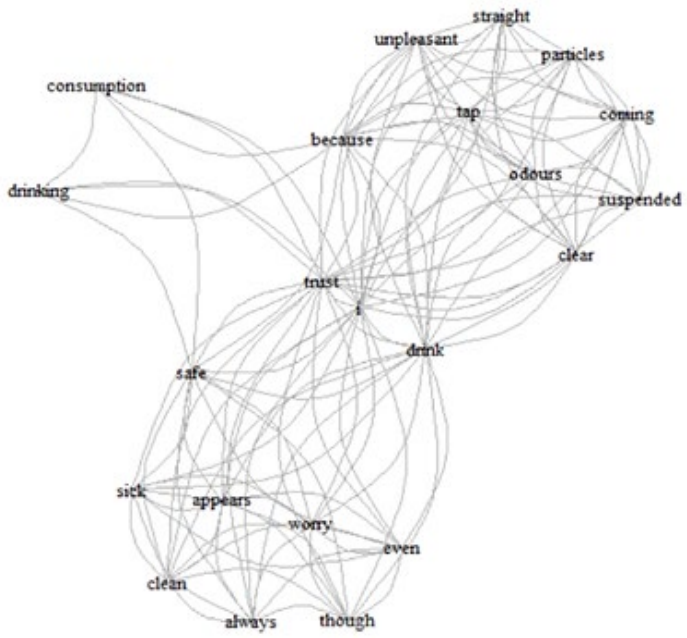

Figure 18 Word network showing connections to words

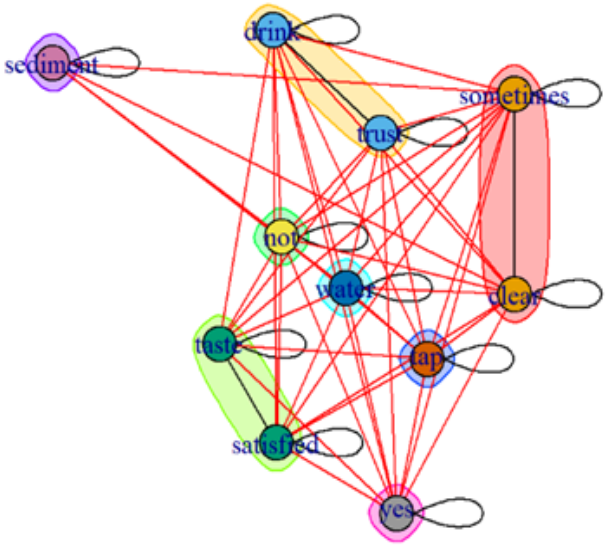

Figure 19 Communities of words

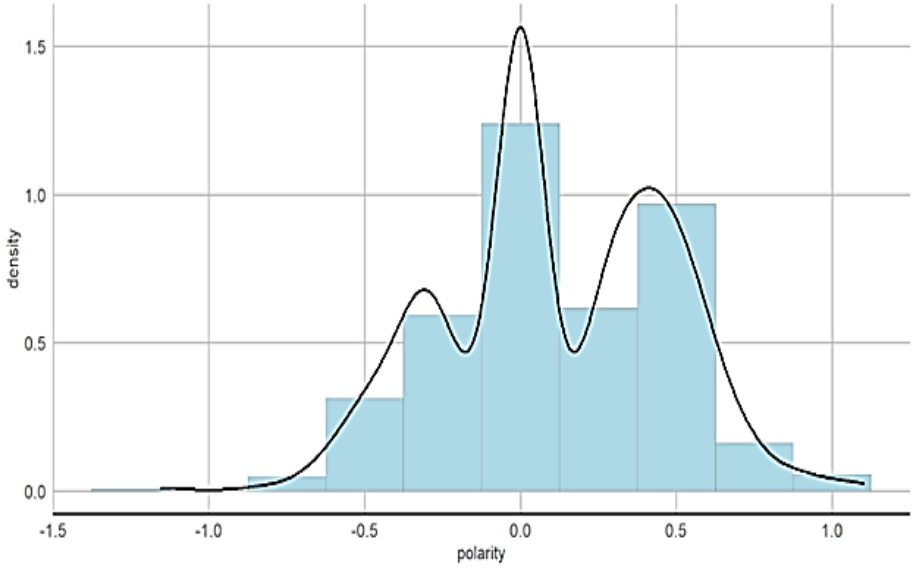

Figure 20 Polarity distribution in sentiment analysis