Text Mining &   
TEXT PROCESSING

Notes compiled by –

Sandeep Kumar Kushwaha

University of Bologna, Italy

Year – 2022.

[Gmail, new, logo Icon in Google new logos](mailto:sandeepkumar998855@gmail.com)

[A picture containing text, clipart

Description automatically generated](https://www.linkedin.com/in/xandie985/)

[Shape

Description automatically generated with low confidence](https://github.com/xandie985)

[Icon

Description automatically generated](https://wa.me/+393517231221)

[](https://www.kaggle.com/sandeepkumarkushwaha)

# ­1.0 Text Processing

## 1.1 Structured Data

Pros:

* Highly organized – SQL is used to manage the structured data.
* Easily decipherable by machine learning algorithms.
* Easily used by business users – it does not require an in-depth understanding of different types of data and how they function. With a basic understanding of the topic relative to the data, users can easily access and interpret the data.
* Accessible by more tools – there are more tools that are available for using and analysing structured data.

Cons:

* Limited usage: Data with a predefined structure can only be used for its intended purpose, which limits its flexibility and usability.
* Limited storage options: Structured data is generally stored in data storage systems with rigid schemas (e.g., “data warehouses”). Therefore, changes in data requirements requires an update of all structured data, which leads to a massive expenditure of time and resources.

Tools:

* [**OLAP**](https://www.ibm.com/cloud/learn/olap)**:** Performs high-speed, multidimensional data analysis from unified, centralized data stores.
* [**SQLite**](https://sqlite.org/)**:** Implements a self-contained, serverless, zero-configuration, transactional relational database engine.
* [**MySQL**](https://cloud.ibm.com/catalog/content/mysql)**:** Embeds data into mass-deployed software, particularly mission-critical, heavy-load production system.
* [**PostgreSQL**](https://www.ibm.com/cloud/learn/postgresql)**:** Supports SQL and JSON querying as well as high-tier programming languages (C/C+, Java, Python, etc.).

Use cases for structured data:

* Customer relationship management (CRM) – CRM software runs structured data through analytical tools to create datasets that reveal customer behaviour patterns and trends.
* Online booking **–** Hotel and ticket reservation data (e.g., dates, prices, destinations, etc.) fits the “rows and columns” format indicative of the pre-defined data model.
* Accounting **–** Accounting firms or departments use structured data to process and record financial transactions.

## 1.2 Unstructured Data

The unstructured data does not have a predefined data model, so it is best managed in non-relational (NoSQL) databases. Another way to manage unstructured data is to use data lakes to preserve it in raw form. Around 80-85% of the data is in the form of unstructured.

Pros

* Native format: Unstructured data, stored in its native format, remains undefined until needed. Its adaptability increases file formats in the database, which widens the data pool and enables data scientists to prepare and analyze only the data they need.
* Fast accumulation rates: Since there is no need to predefine the data, it can be collected quickly and easily.
* Data lake storage:Allows for massive storage and pay-as-you-use pricing, which cuts costs and eases scalability.

*Cons*

* Requires expertise: Due to its undefined/non-formatted nature, data science expertise is required to prepare and analyze unstructured data. This is beneficial to data analysts but alienates unspecialized business users who may not fully understand specialized data topics or how to utilize their data.
* Specialized tools: Specialized tools are required to manipulate unstructured data, which limits product choices for data managers.

*Unstructured data tools*

* [MongoDB](https://www.ibm.com/cloud/learn/mongodb): Uses flexible documents to process data for cross-platform applications and services.
* [DynamoDB](https://aws.amazon.com/dynamodb/): Delivers single-digit millisecond performance at any scale via built-in security, in-memory caching and backup and restore.
* [Hadoop](https://www.ibm.com/cloud/blog/hadoop-vs-spark): Provides distributed processing of large data sets using simple programming models and no formatting requirements.
* [Azure](https://www.ibm.com/cloud/architecture/architectures/ibm-cloud-private-azure/): Enables agile cloud computing for creating and managing apps through Microsoft’s data centres.

*Use cases for unstructured data*

* [Data mining](https://www.ibm.com/cloud/learn/data-mining): Enables businesses to use unstructured data to identify consumer behavior, product sentiment, and purchasing patterns to better accommodate their customer base.
* [Predictive data analytics](https://www.ibm.com/analytics/predictive-analytics): Alert businesses of important activity ahead of time so they can properly plan and accordingly adjust to significant market shifts.
* [Chatbots](https://www.ibm.com/cloud/learn/chatbots-explained): Perform text analysis to route customer questions to the appropriate answer sources.

## 1.3 Semi Structured data

They are partially described by some model, such as hierarchical or graphs.

They don’t obey the tabular structure of data models associated with relational databases or other forms of data tables, but nonetheless contains tags or other markers to separate semantic elements and enforce hierarchies of records and fields within the data.

Examples: emails, XML, RDF, and other markup languages, binary executables.

*Dealing with SSD:*

There are methods and languages to partially deal witch SSD, ex: XPath, XQuery, etc.

## 1.4 The practical implications of Text Mining

* 1. Stock market prediction with 86% accuracy.
  2. Wavii – app for gathering, classifications and distribution of news – acq. By Google.
  3. Summly – news summarizing app of iOS – acquired by Yahoo.
  4. Crime Prediction Systems – CRUSH :is a large database of illegal activities. The goal is to predict the crimes. With the historic data of committed crimes, events, criminals, and behaviour 31% reduction of general crimes was achieved.

## 5 Text Mining

Process of extracting knowledge from unstructured textual data.

*Tasks:*

* Text classifications, clustering & topic modelling
* Text Extraction & Summarization
* Sentiment & opinion mining
* Question answering
* Descriptive Text mining: explaining phenomena from text corpora
* Information retrieval: methods & algorithms to search relevant docs. wrt user queries in the repositories of UD (Unstructured Data).
  + First step for Information retrieval is defining the data mining goals. Then,
  + data selection, where checks are performed to ensure the improved data quality.
  + IR offers efficient methods for representation and selection of UD which can be useful for TM (Text Mining).
  + TM offers techniques to improve complex IR searches.
    - Searching similar docs, within flat or hierarchical catalogues organized by topics.

## 1.6 Representation of Documents

In the matrix of documents and words – we place 1 if document contains the word else 0.

A picture containing shape

Description automatically generated

1.7 Binary Vectors

Vectors 0/1 for each term and document

To find presence of *Bruto*, *Cesare* and *Not Calpurnia* – we take the compliment of *Calpurnia*.

Bruto: 110100 (as it is)

Cesare: 110111 (as it is)

~Calpurnia: 101111 (compliment)

AND op: 100100 (performing AND operation on the three values above)

1.8 Boolean Queries: Exact match

In Boolean Retrieval model, the query expressive power is based on Boolean propositions.

These queries consist of AND, OR & NOT

* Each doc is considered a set of terms
* Matching is strict – Binary (Yes/No)

Simplest form of IR.

It’s predominant solution even in the modern software.

1.9 Limits of the Vector Representation

Let’s suppose, the number of documents is N = 1 million. Each of the document contains 1000 terms.

* If each term takes 6 bytes, we will have 6 GB of documents.

Let's suppose, the number of distinct terms M = 500k in million of documents.

500k X 1M is matrix with 500 billion 0 and 1 as values. (at most 1 billion of values as 1)

Sparse matrix of 1000 terms \* 1 M documents.

A better way of representation?

1.10 Inverted index

We store the terms and their occurrence in the form of dictionary, where each term t is stored as key, whereas fixed sized list of all the documents containing t is stored as value of the key. Each document is identified by docID.

*Bruto*  🡪 [1,2,4,11,21,33,45]

*Cesare* 🡪 [1,2,3,6,34,132,-]

*Calpurnia* 🡪 [2,32,55,121,-,-,-]

*keys* on RAM 🡪 *values* are stored on disk

1.10.1 Boolean queries with Inverted Index

We perform AND operation on lists included in the query. So, *Bruto* AND *Cesare* AND ~*Calpurnia.*

1.10.2 Query Optimization: Example

We can visit the lists in increasing order of frequency (no. of docs containing the term).

Starting from the shorter list for better efficiency.

So, we start from *Calpurnia 🡪 Cesare 🡪 Bruto*.

1.10.3 Arbitrary Boolean Queries

How to process more complex queries?

Like, (*Bruto* OR *Cesare*) AND NOT (*Antonio* OR *Cleopatra*)

**Naïve method:** Lista(*Bruto*) U Lista(*Cesare*) – Lista(*Antonio*) U Lista(*Cleopatra*)

(*Bruto* AND ~*Antonio* AND ~*Cleopatra*)OR(*Cleopatra* AND ~*Antonio* AND ~*Cleopatra*)

the union produces usually lists that are longer than intersection, therefore it is more efficient usually to process intersection before union.

1.10.4 Building the Inverted Index

1. Collect the documents to be indexed:

\framebox{\weestrut Friends, Romans, countrymen.} \framebox{\weestrut So let it be with Caesar}

1. **Tokenize the text**, turning each document into a list of tokens:

\framebox{\weestrut Friends} \framebox{\weestrut Romans} \framebox{\weestrut countrymen} \framebox{\weestrut So}….

1. **Linguistic pre-processing**, producing a list of normalized tokens,

\framebox{\weestrut friend} \framebox{\weestrut roman} \framebox{\weestrut countryman} \framebox{\weestrut so} ...

1. **Index the documents** so that each term occurs in by creating an inverted index, consisting of a dictionary and postings.

\begin{figure}
% latex2html id marker 1068
\begin{tabular}{p{2.3in}p{2.6in}}
\te...
...n each document) or the position(s) of the term in each
document.}
\end{figure}

X. Inverted Index: RDBMS vs NoSQL DataBase

**Generalized Inverted Index (GIN)**

* contains an index entry for each term, with compressed list of matching locations.
* Best for static data.
* Lookups are faster.

**Generalized Search Tree (GiST)**

* generalization of several traditional indexes, like B-trees, with no limitation in text size, moreover it allows using arbitrary predicates.
* Best for dynamic data.
* Data update under 100k terms is faster.

DBMS systems are equipped with such indexes for increasing the speed of full text searches.

* NoSQL is used for text manipulation, but it is less efficient than RDBMS.

X.Text Tokenization

Tokenization is essentially splitting a phrase, sentence, paragraph, or an entire text document into smaller units, such as individual smaller words/ terms, called tokens.

Challenges in tokenization:

* India’s 🡪 India/Indias/India’s?
* U.S. 🡪 U.+ S. breaking up?
* Co-author 🡪 shall we divide the terms?
* Date formats
* Language issues:
  1. Chinese & Japanese don’t use spaces to separate words.
  2. German has long composed nouns without separations.
  3. Arabic & Hebriw are written R to L, but not in case of numbers.

X. Stop Words

Frequent occurring words carrying very little information.

General stop word list, and domain specific stop words are also there.

X. Word Normalization

Normalization is **the process of converting a token into its base form**. In the normalization process, the inflectional form of a word is removed so that the base form can be obtained.

X. Language dependent Normalization and Tokenisation

Both of them depend of the language. But there are cases where multilingual usage is present, how to handle such cases?

Search Expansion: users give additional input on query words or phrases, possibly suggesting additional query terms.

Uppercase and Lowercase: FED/SAIL 🡪 fed/sail which is totally different from contrast.

Synonyms and phonetic equivalence: Soundex

car = automobile color = colour

Term equivalence by phonetic heuristics

Developed by international police dept. to unify wanted criminal names differently registered in different countries. *Hermann* as *Herman* etc.

Generate for each term phonetic hash so that terms with a similar sound have same code – spandex alog