**Data Mining**

Armin Pousti
Dmitrii Vlasov
Mcgill Department of Computer Science

COMP 480
Professor: Joseph Vybihal
June 14, 2023

As the constant and quick development of social networks continues, it is difficult to handle the great pool of unstructured data available in different fields and domains, including business, governments, and well-being. These unstructured data can have different forms, such as text, image, audio, and video data. Data mining extracts information, patterns, and knowledge from these immense and massive amounts of datasets. Finding the data from large databases is an essential step in data mining. The three main challenges in handling social network data are size, noise, and dynamism. With the advancement of computer technology and storage chips that have become cheap and extensive, people are collecting various types of data for different purposes. To deal with the chaotic and overwhelming situation of dealing with this vast data, it introduced the notion of database management systems (DBMSs). Using efficient methodical approaches, DBMSs can create databases, manage data, and retrieve extensive information collection. Data mining is an intelligent concept of sifting through historical data to uncover relevant information.

Text mining is one of the most powerful methods to analyze and process unstructured data. Since data can be concurrently getting bigger, processing, analyzing, and storing these enormous volumes of data would become a challenging task. Text analytics, or text data mining, extracts high-quality textual information and insights from large volumes of unstructured text data. There are a few general steps taken for the text data mining process. First, the text miner collects unstructured data from various resources, including web pages and files in pdf, plain text, or emails. After that, the data pre-processing and cleansing begin, which detects and removes anomalies from the collected data using various text mining tools. The text miner then converts the extracted information into a structured format. The patterns within the dataset are analyzed, and the text miner stores all the valuable and relevant information in a secure database.

There are several techniques that text miners use in the field of natural language processing to understand and analyze human language. One of the best analysis techniques (and the one that is used in the YouTube comments analysis in the Python script accompanying this report) is sentiment analysis. Sentiment analysis, also known as opinion mining, is a machine learning and natural language processing technique used to examine the emotional tone of a text. This technique is commonly employed by businesses to evaluate the sentiment associated with their brands, products, or services. Customer feedback analysis is a prevalent application of sentiment analysis, allowing companies to assess and analyze customer emotions and opinions. By identifying the underlying motives and emotions behind purchases or click-throughs, sentiment analysis provides valuable insights and the ability to predict future responses and adjust marketing and advertising strategies.

In sentiment analysis, sentiments are categorized into polarities such as positive, negative, or neutral. As such, one can quantify the positivity or negativity of the text. The process typically involves the following steps:

1. Text preprocessing: Text is cleaned by removing unnecessary characters, punctuation, and common stopwords (such as “the” “and” “this” etc, similar to the TF-IDF model described in the previous report). Then the remaining words are transformed into their base or root form and then the text is tokenized.
2. Feature extraction: Relevant features or attributes are extracted from the preprocessed text. The following is a list of common feature extraction techniques:

    *Bag-of-Words* - representing the text as a collection of individual words without considering their interconnectivity (isolating words like “fantastic”, “awful”, and “confusing”)

    *n-grams* - capturing sequences of n consecutive words as features. For example, the sentence “The plot was confusing” would yield a tri-gram “plot was confusing” as a single feature.

*Part-of-Speech* - assigning grammatical tags to each word such as nouns, verbs, adjectives etc. This simplifies the identification of sentiment-bearing words such as positive adjectives like “fantastic” or “perfect”.

3. Sentiment Classification: Machine learning models such as *Naive Bayes* are trained on labelled datasets to classify text into different sentiment categories
4. Sentiment Analysis: The trained model is then used on an unlabeled text to predict the sentiment category based on the learned patterns and associations.

The above steps produce the sentiment scores which indicate the following:

- neg: negative sentiment score (0 to 1), where 0 means no negative sentiment
- pos: positive sentiment score (0 to 1), where 0 means no positive sentiment
- neu: neutral sentiment score (0 to 1), where 0 means no neutral sentiment
- compound: compound score (-1 to 1) which represents the overall sentiment of the text, where scores less than 0 indicate negative sentiment, scores greater than 0 indicate positive sentiment and scores around 0 indicate neutral sentiment

For example, consider three movie reviews:

"The movie was an agonizing torture of incoherent storytelling and mind-numbingly dull characters. I would rather endure a root canal without anesthesia than suffer through that cinematic abomination again."

"This movie is pure cinematic magic, an unforgettable masterpiece that redefines the very essence of cinema itself."

"The movie was okay, neither particularly exciting nor overly disappointing. It was just average and didn't leave a lasting impression."

These texts have sentiment scores of -0.8151, 0.6249, and 0.0426 respectively.

Another data type that is used for data mining is images and videos. Images have become a popular way to preserve memories, with applications in sports, medicine, and social networking. Image processing has gained acclaim as a research field, involving the acquisition

of images and extracting valuable information from them. Image mining focuses on discovering implicit knowledge, relationships, and patterns in images. One possible application of image mining is the use of such techniques in the field of medicine where medical professionals regularly capture and store large amounts of medical images such as X-rays and MRI scans. Image mining can potentially assist in the automated detection and diagnosis of diseases by analyzing patterns and anomalies within medical images.

Video mining extends the concept to videos, discovering features within the video data. Algorithms divide the video into smaller segments based on changes in visual content (typically at scene boundaries) and then extract meaningful features from each shot or frame. These features often include color histograms, textures, and shapes which all capture the visual characteristics of the video. The extracted features are then analyzed for patterns. As an example, these patterns can then be used to identify objects, monitor crowd behavior, or look for anomalies to detect suspicious activity.

Cluster analysis, also known as clustering, is a technique used in data analysis that organizes objects or data points into several groups based on similarity by some means. Objects in each cluster are similar and dissimilar to objects in other clusters. Cluster analysis aims to identify patterns or structures in groups of clusters within multi-dimensional datasets based on some similar attribute. A cluster center, called a centroid, usually resembles each cluster; therefore, Euclidean distance is one good measure used in machine learning and pattern recognition for cluster analysis. Various fields use clustering because cluster analysis enables researchers and practitioners to discover patterns, identify structures, and gain insights from data to make informed and appropriate decisions.

Classifying data in data mining is typical by applying different classification methods. The goal of classification is to compute each existing class variable's value accurately. The classification process is divided into two steps. In the first step, the learning step, sample data is selected randomly from the dataset. In the second step, the classification step verifies its accuracy after

assigning the data values to the model. Different clustering algorithms include hierarchical clusters, K-means, OPTICS, DBSCAN, Cobwebm, and EM, in which the K-mean algorithm outperforms the other algorithms and is the easiest to use.

The objective of the clustering algorithm is to successfully and effectively process a wide variety of data and bring down the volume of data by segregating similar data, which helps establish taxonomies or categories.

Clustering is grouped into two categories, hard clustering, and soft clustering.

In hard clustering, also known as exclusive clustering, each data point is either entirely placed into a cluster or not, so it has a definitive membership. There are no overlaps, and each data point is exclusively assigned to a single cluster which partitions the data into distinct, non-overlapping clusters. Based on each data point's similarity or proximity to the data points in a cluster, the data point is assigned to the cluster with the most significant similarity measure. The most commonly used algorithm for hard clustering is K-means. In K-means clustering, data points get assigned to a cluster with the nearest centroid to that data point. Hard clustering is used when clear separation and distinctiveness between clusters are desired.

In soft clustering, also known as fuzzy clustering or probabilistic clustering, each data point is not put into a separate cluster. However, each data point receives a probability or likelihood for it to be in those clusters. In soft clustering, data points can belong to multiple clusters simultaneously. In other words, in soft clustering, each data point receives a degree of membership to belong to each cluster. The Gaussian mixture model is one example of soft clustering.

Another clustering algorithm technique is hierarchical clustering. In hierarchical clustering, a top-down hierarchy of clusters is used. Based on the hierarchy, the data objects are decomposed into clusters. There are two approaches followed to create clusters: top-down and bottom-up. The top-down approach, called divisive clustering, starts with all data points in a single cluster and then splits the clusters into smaller clusters in succession using a dissimilarity

measure until the algorithm achieves the termination condition, which can be either reaching the desired number of clusters or that there is no more meaningful division to be made. The bottom-up approach, called agglomerative clustering, starts by treating each data point as a separate cluster and then progressively merging the most similar clusters using a similarity measure in succession until it reaches a predefined number of clusters or all clusters are merged into one.

To understand various data mining techniques comprehensively, we conducted analyses on two datasets: one obtained from the YouTube API and the other from COVID-19 data in Canada. The COVID-19 dataset was subjected to multiple analytical approaches, including cluster analysis, correlation analysis, trend analysis, and EVA (further elaborated in subsequent sections of this report) to identify underlying patterns in the data. Conversely, while performing numerical analysis on the YouTube dataset, the primary focus was text mining techniques. Specifically, sentiment analysis served as the core algorithm employed in this aspect of the project. Additionally, the utilization of the YouTube API also acts as a stepping stone to the subsequent module on collaborative filtering.

---

## Data Mining on COVID-19 Data

### Preprocessing

COVID-19.py data mining python script starts by fetching COVID-19 data from multiple API endpoints for data on the number of deaths, cases, completed tests, and hospitalization of COVID-19 in Canada daily and in total up to that date. It then creates a data frame for each of the data sets. After cleansing these individual data frames from unnecessary columns and preprocessing them, it merges all the individual data frames on the number of deaths, cases, completed tests, and hospitalization into a vast data frame using the date, which will contain all the necessary data to be analyzed.

**Exploratory Data Analysis**

After preprocessing the data into a single data frame, the data is ready for further analysis. One way to analyze our information is to run Exploratory Data Analysis (EDA) on the data frame. It calculates the data frame's summary statistics, including metrics such as count, mean, standard deviation, minimum, quartiles, and maximum values for each numeric column. This information provides insights into the central tendency, dispersion, and distribution of the numeric data in the data frame. Exploratory Data Analysis provides an initial exploration of the data to understand its structures. The code produces line plots to identify data trends, patterns, and outliers for better visualizations as well as printing the information as a data frame.

**Stats plot**

The code allows the user to create line plots to compare the daily or total cases, deaths, and completed tests from a given data frame over time. These graphs visually represent the trends and patterns in the data providing a visual analysis to the user. The user can select between the daily and total representation in the user interface.

**Trend Analysis**

The code also can perform trend analysis on the processed data frame. Using trend analysis, we can track trends in COVID-19 cases, deaths, hospitalizations, and testing over time. By creating line plots or time series, we can observe the progression of the pandemic in Canada, identify its peaks, understand the effectiveness of interventions, or predict possible future trends. The code calculates the 7-day rolling average for the specific variable and plots it next to the daily values to allow for a visual assessment of the general direction of fluctuations in the variable, which helps the user understand the trend over time.

**Correlation Analysis**

The code also provides a correlation analysis of the data frame. The correlation analysis examines the relationship between the daily and total cases, deaths, completed tests, and hospitalization in the data frame. It calculates the correlation coefficients to determine the

correlation between two variables and the strength of the correlation. This analysis helps identify factors that influence the spread or impact of the virus; it also helps in understanding the relationships between variables and identifying potential patterns or dependencies within the data. To visually represent the correlation analysis, first, we have to compute a correlation matrix that measures the pairwise correlation between the columns of the data frame to provide insights into the relationship between the variables. Then, the code generates a heat map plot of the correlation matrix. Each cell represents the correlation between two variables in the heat map plot. The values are annotated in the cell, indicating the strength and direction of the correlation. There is also color to show the strength of the correlation. The more red a cell, the closer to 1 the correlation, and the more blue the cell, the closer to -1 the correlation. So the heatmap allows for easy identification of strong positive or negative correlations between variables.

**Cluster Analysis**

One other data mining analysis is clustering analysis to analyze the COVID-19 data. The code performs clustering analysis using the K-means clustering on the data frame. The K-means clustering partitions the dataset into the number of clusters specified by the user, where each data point belongs to the cluster with the nearest mean or centroid, as discussed in the earlier part of this report.

The algorithm has five stages:

**1) Initialization**: The algorithm randomly selects K points from the dataset as initial centroids, which is also the desired number of clusters

**2) Assignment**: The algorithm assigns the remaining data points to the nearest centroid based on a distance metric, commonly the Euclidean distance.

**3) Update**: The algorithm then recalculates the centroid of each cluster by taking the mean of all data points assigned to that cluster.

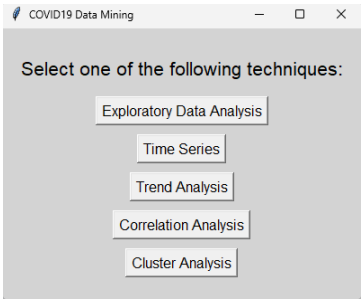
**4) Repeat**: The algorithm keeps assigning and updating until the centroids no longer change significantly

**5) Convergence**: The algorithm converges when the centroids stabilize, meaning that the data points no longer get assigned

The result is the final K clusters, where each data point gets assigned to a specific cluster. K-means clustering is a simple technique for exploring patterns for identifying natural groupings or clusters. The code adds a 'Cluster' column that indicates the assigned cluster for each data point. The visualization step uses the 'Cluster' column. For visualizing the cluster analysis. The code also adds a scatter plot to represent the clusters where each data point is colored based on its assigned cluster, creating a visual separation of the clusters. This visualization can provide insights into the grouping and distribution of data points within different clusters, which allows for examining patterns or relationships within the data based on the identified clusters.

**GUI**

For a better interactive mode for the user to use the different techniques provided by the COVID19.py script, a graphical user interface is provided to select the method they want the data to be analyzed with and get their desired graphical representation.



## COVID19.py Documentation

```
def preprocessing():
    """
    Preprocessing retrieves data on deaths, cases, completed tests, and hospitalization for COVID-19in Canada from an
    API endpoint. It creates a data frame for each of the data sets, and after cleansing the individual data frames
    from unnecessary columns, it merges all the data frames to create a huge data frame containing all the necessary
    information on COVID19 data.
    :return: <class 'pandas.core.frame.DataFrame'> Returns the processed DataFrame (df_combined)
    """
def perform_EDA(data_frame):
    """
    This function performs Exploratory Data Analysis (EDA) on a given data frame. It calculates the data frame's
    summary statistics, including metrics such as count, mean, standard deviation, minimum, quartiles, and maximum
    values for each numeric column. Prints the summary Statistics and returns it as a data frame. The user can save
    it as an Excel file as well.
    :param data_frame: <class 'pandas.core.frame.DataFrame'> The processed DataFrame on COVID19 data
    :return: It does not return anything, and will just plot the EDA on the graph
    """
def stats_daily_plot(data_frame):
    """
    This function creates a line plot to compare the daily cases, deaths, and completed tests from a given data frame
    over time.
    :param data_frame: <class 'pandas.core.frame.DataFrame'> The processed DataFrame on COVID19 data
    :return: It does not return anything, and will just plot the daily stats graph
    """
def stats_total_plot(data_frame):
    """
    This function creates a line plot to compare the total cases, deaths, and completed tests from a given data frame
    over time.
    :param data_frame: <class 'pandas.core.frame.DataFrame'> The processed DataFrame on COVID19 data
    :return: It does not return anything, and will just plot the total stats graph
    """
def trend_analysis(df, variable):
    """
    The function calculates and displays the trend of the specified variable over time by plotting the daily values and
    a 7-day rolling average
    :param df: <class 'pandas.core.frame.DataFrame'> The processed DataFrame on COVID19 data
    :param variable: <String> Daily/total cases, deaths, completed tests, and hospitalization for the column to analyze
    :return: It does not return anything, and will just plot the daily tren analysis graphs
    """
```

```
def correlation_analysis(df):
    """
    The function visually represents the correlation between numeric variables in the data frame
    :param df: <class 'pandas.core.frame.DataFrame'> The processed DataFrame on COVID19 data
    :return: It does not return anything, and will just plot the daily trend analysis graphs
    """
def cluster_analysis(df, num_clusters):
    """
    Applies K-means clustering to the numeric columns of the DataFrame, grouping similar data points into clusters.
    :param df: <class 'pandas.core.frame.DataFrame'> The processed DataFrame on COVID19 data
    :param num_clusters: <int> for the user to indicate the number of clusters
    :return: <class 'pandas.core.frame.DataFrame'> COVID19 processed DataFrame with the addition of 'Cluster' column
    """
def visualize_clusters(df, variable):
    """
    the code helps visualize the clusters obtained from the clustering analysis by plotting the data points on a
    scatter plot, where each point is colored based on its assigned cluster
    :param df: <class 'pandas.core.frame.DataFrame'> COVID19 processed DataFrame with the addition of 'Cluster' column
    :param variable: <String> Daily/total cases, deaths, completed tests, and hospitalization for the column to analyze
    :return: It does not return anything, and will just plot the cluster analysis graphs
    """
```

**User_interface**

Also, the command line user interface allows users to create a graphical representation of the technique they used to analyze the COVID-19 data.

An example of user input is as follows:

```
COVID19 Data Mining
Select one of the following techniques:
1) Exploratory Data Analysis
2) Time Series
3) Trend Analysis
4) Correlation Analysis
5) Cluster Analysis
Option : 5
Choose the number of clusters : 4
Selected one of the following options:
1) Daily
2) Total
Option : 1
Select the data set you want to analyze:
1) Daily Cases
2) Daily Deaths
3) Daily Completed Tests
4) Daily Hospitalization
Option : 2
```

---

**Data Mining YouTube statistics**

**main**

It is worth noting that running main requires changing the value of *api_key* variable using one's own YouTube API key, however, other modules can still analyze already gathered results without it.

1) Video Search

```
search_response = youtube.search().list(

q='YOUR_TOPIC',

part='id',

maxResults=50
```

```
).execute()

video_details = []

for item in search_response['items']:

video_id = item['id'].get('videoId')

if video_id:

video_response = youtube.videos().list(

id=video_id,

part='snippet,statistics'

).execute()

video_details.append(video_response['items'][0])
```

Performs a search query on the YouTube API with the query term *YOUR_TOPIC*. Using the snippet and statistics parameter, the video details are appended to the *video_details* list.

2) Handling multiple pages

```
page_count = 1

while 'nextPageToken' in search_response:

next_page_token = search_response['nextPageToken']

search_response = youtube.search().list(

q='music in english',

part='id',

maxResults=50,

pageToken=next_page_token

).execute()

for item in search_response['items']:

video_id = item['id'].get('videoId')

if video_id:

video_response = youtube.videos().list(

id=video_id,
```

```
part='snippet,statistics'

).execute()

video_details.append(video_response['items'][0])

page_count += 1

if page_count >= 3:

break
```

By default, the query to the API can only fetch one page of YouTube videos (up to 50 videos). Therefore, if there are additional pages of search results, the code retrieves the *nextPageToken* from the previous search response and uses it in the subsequent search request to fetch the next set of videos.

3) Fetching Comments

```
for video in video_details:

video_id = video['id']

try:

comments_response = youtube.commentThreads().list(

part='snippet',

videoId=video_id,

maxResults=50

).execute()

video['comments'] = []

if 'items' in comments_response:

for comment in comments_response['items']:

comment_text = comment['snippet']['topLevelComment']['snippet']['textDisplay']

video['comments'].append(comment_text)

except HttpError as e:

if e.resp.status == 403 and b'commentsDisabled' in e.content:

# vdeo has disabled comments
```

```
video['comments'] = []
```

Due to the limitations of the API, an additional API request needs to be done individually for each video to append comments to the csv file later. For each video in the *video_details* list, the code tries to fetch comments. The try-except is used to catch the HttpError exception that is caused by the video having comments disabled.

4) Saving data to the csv file

**analysis**

The modules analysis.py and sentiment_analysis.py perform some basic inquiries into the data like calculating average numeric metrics, finding the most viewed, liked, and commented video in the category, calculating correlation (to find whether likes or comments are more important to the view count), as well as creating plots to visualize the best-performing videos. It is worth noting that the graphs provide rough information on the videos since older videos may potentially perform better than new ones because such videos had more time to grow in views, likes, and comments. A potential improvement may be an implementation that divides the metrics by the number of days the video was out in order to obtain normalized metrics.

**sentiment_analysis**

Finally, using the *nltk.sentiment* natural language processing module the sentiment analysis explained earlier is performed on the mined videos. The code appends the sentiment score to the csv file to allow further exploration of video features.

The acquisition of experience in the use of YouTube API provides a great opportunity to use it in the next module on collaborative filtering.

## Work Cited


- Najafabadi, Maryam Khanian, et al. “A Survey on Data Mining Techniques in Recommender Systems - Soft Computing.” *SpringerLink*, 7 Nov. 2017, link.springer.com/article/10.1007/s00500-017-2918-7.
- “Big Data and Social Media Analytics.” *SpringerLink*, link.springer.com/book/10.1007/978-3-030-67044-3. Accessed 15 June 2023.
- “An Introduction to Data Mining in Social Networks.” *Advanced Data Mining Tools and Methods for Social Computing*, 4 Feb. 2022, www.sciencedirect.com/science/article/abs/pii/B9780323857086000084.
- Robillard, Martin P., et al. An Introduction to Recommendation Systems in Software Engineering. Springer Science and Business, Chapter 3, 2014.
- Robillard, Martin P., et al. An Introduction to Recommendation Systems in Software Engineering. Springer Science and Business, Chapter 6, 2014.
- Robillard, Martin P., et al. An Introduction to Recommendation Systems in Software Engineering. Springer Science and Business, Chapter 7, 2014.