Universitetet i Sørøst-Norge

Analysis of YouTube Trends to Map Societal Interests

CS4010: Data Management Course

8552

8560

8581

Department of Science and Industry Systems

Faculty of Technology, Natural Sciences and Maritime Sciences

University South-Eastern Norway, 2024

Abstract

This case study describes the engineering of an infrastructure mostly built as containers that stores and analyzes statistical data from YouTube's Application Programming Interface (API) using Big Data technologies such as Apache Spark for data processing, MongoDB for data storage, and Microsoft Power BI for data visualization.

Keywords: Apache Spark, Docker, MongoDB, Microsoft Power BI, data extraction, data transformation, data loading, data analysis, data visualization

Preface

Contents

Abstract **i**

Preface **ii**

1 Introduction **1**

1.1 Motivation 1

1.2 Problem Statement 2

1.3 Hypothesis 2

1.4 Project management 3

2 Theoretical Background **4**

2.1 Literature Review 4

2.2 Technologies and Standards 5

2.2.1 Justifying Selection of Tools and Technologies 7

3 Method **8**

3.1 System Architecture 8

3.2 Data Extraction 10

3.3 Data Transformation and Loading 12

3.3.1 YouTube Trends 13

3.3.2 Google Trends 16

3.3.3 YouTube Word Collection 17

3.3.4 Google Trends Word Collection 18

3.3.5 Data Visualization - Power BI 18

4 Results **21**

4.1 Google Trends Data 21

4.2 YouTube Words Collection 21

4.3 YouTube Trends Collections 22

4.4 Analyzing Data Using Power BI 23
4.4.1 Tags and Keywords 23
4.4.2 Biggest positive change in YouTube views 23
4.4.3 Video Categories 24
4.4.4 Google Trends 24

5 Discussion & Conclusion **26**
5.1 Challenges Faced 26
5.2 Discussion 28
5.3 Conclusion 30

References **31**

Chapter 1

Introduction

We are interested in studying the societal impact of YouTube, particularly if the virality of its videos are a reflection of societal concerns and interests. By conducting a big data analysis, we aim to observe potential correlations between the current societal trends and the traction of YouTube videos. *By "trends" or "societal interests", this report refers to quantitative factors on YouTube and Google that acts as indicators of popularity within a given region.* Examples include most popular searches on Google, and the propagation of a news-worthy event discovered on Google to achieve prevalence on YouTube. This case study has chosen Norway as the scope of this analysis.

1.1 Motivation

Big data analyses on YouTube exist [see Subsection 2.1]. However, significant research gaps remain, particularly regarding YouTube Shorts—brief (≤ 3 minutes) and self-looping videos similar to TikTok [1], [2]. Including YouTube Shorts in our data collection enables us to examine how their introduction has transformed YouTube's role as a media platform, particularly in societal interests and screen retention.

Another overlooked aspect in previous studies is the use of quantitative data to identify correlations between YouTube trends and regional events. For example, after a football game is featured in the news, it may seem intuitive for viewers to turn to YouTube for game highlights. The same applies to political news and other significant events. This study is significant as it explores whether a symbiotic relationship exists between newsworthy events and the popularity of YouTube videos and their categories.

1.2 Problem Statement

YouTube is the second most popular social network by active users, with 95% of the global internet population watching its content [3]. These users display significant regional and interest-based diversity [3], supported by YouTube's 32 distinct video categories [4]. Analyzing YouTube data can yield valuable insights into what engages various demographic groups and geographic regions, as well as how these interests evolve over time. For instance, trends in video length could shed light on attention spans, while viral content may reflect societal interests.

To effectively analyze trends, societal behaviors, and interests, access to diverse and high-quality data is essential. The volume, variety, velocity, and veracity of the data determine the potential value of such analyses [5]. This project seeks to uncover key societal trends in Norway by leveraging YouTube metadata, offering actionable insights for marketers, researchers, and policymakers.

Google is a widely used search engine for discovering newsworthy events [6]. It provides historical insights into search activity through its platform, Google Trends. This paper aims to examine the correlation between societal interests in Norway, as represented by Google Trends data, and the virality of YouTube videos and categories.

Problem Statements:

1. *How can metadata from YouTube's API be analyzed to uncover and understand societal interests over time across demographics and regions?*
2. *What (if any) is the correlation between societal interests and behaviors, and the virality of YouTube videos and categories?*

1.3 Hypothesis

We hypothesize that shorter videos (0–3 minutes) will garner the highest number of views, with the exception of podcasts. This expectation is informed by our literature review 2.1 indicating that viewers tend to have short attention spans [7]. Consequently, we anticipate

that YouTube Shorts will constitute 30% or more of the platform's most trending videos. Additionally, we hypothesize a strong positive correlation (Pearson $r > 0.5$) between societal trends, as represented by Google Trends data, and the popularity of YouTube videos and categories. This hypothesis is based on empirical evidence suggesting a link between Google search activity and YouTube viewership [7].

1.4 Project management

8552: Data extraction using Google's API, data transformation of Google Trends data using Apache Spark, and data loading using MongoDB.
8560: System integration using Docker, data transformation of YouTube Trends data using Apache Spark, and data loading using MongoDB.
8581: Data visualization and analysis using Microsoft Power BI.

For a detailed description of each student's contribution, please refer to the Acknowledgments appendix.

Chapter 2

Theoretical Background

2.1 Literature Review

A 2021 case study in Frontiers in Applied Mathematics and Statistics analyzed YouTube as a learning platform, focusing on a channel providing hand surgery educational content [7]. It concluded that videos should be no longer than two minutes since "there is a decline in engagement with an increased length of the video" [7]. This suggests YouTube Shorts may be ideal for educational resources due to their brevity. However, their self-looping nature makes them less suitable for self-paced learning, which the study emphasized as important [7].

The study examined metrics like watch time, views, likes/dislikes, subscribers, and demographics, finding that 33.3% of traffic came from YouTube searches, 19.4% from suggested videos, and 19% from external links. Notably, 57% of external traffic originated from Google searches [7], hinting at a possible correlation between trends on Google and YouTube. The report also discusses three multimedia learning principles; two of which we consider relevant to our case study.

1. **The limited capacity principle**: This principle assumes that "individuals have a limited ability to absorb information at any one time" [8]. It challenges the idea that videos should always be as short as possible, particularly if brevity hinders comprehension. This might explain the fast-paced nature of YouTube Shorts and their potential limitations for detailed or nuanced content. Conversely, it may suggest long-form podcasts as an outlier, since they deliver information in a slower, more digestible format.

2. **active-processing principle**: According to this principle, "people should be actively engaged in the learning process rather than passive receivers of information" [8]. While this principle was developed for educational content, we question whether it applies to other categories of videos. Engagement methods vary—educational videos might prompt viewers to pause and solve a problem, while other videos may rely on captivating titles, such as "Can YOU spot (X)?" to stimulate interest [see Figure 2.1].

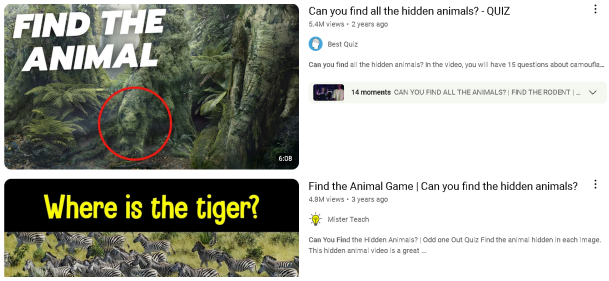

Figure 2.1: Screenshot showing popular videos of type: "Can YOU spot (X)?

2.2 Technologies and Standards

- **Apache Spark**: "Apache Spark is a unified analytics engine for large-scale data processing. [9]". It allows data engineers to create SQL queries on data that is not in an SQL database, either through Spark SQL, or with programming languages such as Python, Scala, Java and R [10].

- **File Formats: CSV** which is short for Comma Separated Values is a standard file format with the first line, or the header, describes what the values for the subsequent lines represents. **JSON** is a standard file format that is both readable for humans and computers, and provides the ability to share data between different processes [11].

- **ISO-8601** ISO-8601 is an international standard format that helps in relaying information regarding dates and times, with the order of dates formatted as: YYYY-MM-DD [12]. For duration, the same order of magnitude applies. However, a 'P' is prepended to the duration string which is used to describe a period between years and days, and a 'T' is used to as a separator to describe time between hours and seconds [13].

- **Microsoft Power BI Desktop** Power BI is a data visualization tool that can import data, or even connect to data sources such as MongoDB through a pre-established interface, also referred to as a connector. It can be used to transform and visualize data that can later be used for data analysis which helps analysts in presenting and relaying the data in a higher level with colored dynamic graphs [14].

- **MongoDB** MongoDB is a NoSQL database management system (DBMS) designed to store and manage large volumes of data in a flexible, scalable way. Unlike traditional relational databases that use tables and rows and SQL to query these, MongoDB stores data in JSON-like documents, which are organized into collections. This document-oriented approach allows for more dynamic and varied data structures, making it ideal for applications that require fast, iterative development and handling of diverse data types without the need for SQL (NoSQL) [15].

- **MongoDB Connector for BI**

  Tool needed for PowerBI and MongoDB server to communicate with each other. It functions as a translation layer in-between MongoDB and PowerBI by converting SQL queries made in Power BI into MongoDB queries that can be understood by the database server [16].

- **Docker** Docker is a tool for quickly building, creating and starting services that would otherwise need to be installed for each developer's host machine. Docker simplifies this process by either using pre-configured and pre-built **Docker Images** or using a Docker Image as a template for creating your own Docker Images. Each Docker Container can be isolated and does not have default access to your file system or data, though this is possible [17]. Since Docker normally runs as the root user it does introduce security risks [18].

- **Python Packages**

  - The *PySpark* Python package is the Python API for Apache Spark, which is a unified analytics engine for large-scale data processing and analysis. PySpark supports key Spark features like Spark SQL for SQL and structured data processing, pandas API on Spark for pandas workloads, MLlib for machine

learning, Structured Streaming for incremental computation and stream processing, and Spark Core [19][9].

- The *PyMongo* Python package enables connection to and interaction with MongoDB. It provides tools to perform database operations like inserting, querying, updating, and deleting documents within MongoDB collections. PyMongo is the recommended way of working with MongoDB databases with Python [20][21].
- The *google-api-python-client* Python package provides the means to establish an API client to query Google's many API's. It enables easy handling of authentication and simple API calls. The package is maintained by Google and is available via the pip package manager [22][23].

2.2.1 Justifying Selection of Tools and Technologies

The selection of technologies and standards was informed by our lecture week and coursework, where we gained foundational experience with these tools. While Apache Kafka was initially considered for transferring data between the data extraction and transformation scripts, it was deemed unnecessary since data extraction for Google Trends could not be automated, and Apache Spark was capable of processing CSV and JSON files directly. For other technologies, alternatives were not extensively explored, as the chosen tools were sufficient to build a functional prototype. With a broader project scope, more time would have been allocated to evaluating additional options.

Chapter 3

Method

3.1 System Architecture

In Figure 3.1, we have illustrated three separate Docker Containers. The first container highlighted in blue from the left is the Transform and Load Docker Container, which can now be defined as the ETL container since this container is now responsible for data extraction, transformation, and loading. The second container highlighted in blue in the middle, is the Docker Container which contains the MongoDB server. This container is connected to the ETL container through a bridged Docker Network, later described in Fig. 3.2. The third and final container is the mongodb-bi-connector. This container allows Microsoft Power BI to connect to a MongoDB server, and use the data in the existing databases on this server for data presentation, visualization, and hence, data analytics. For the MongoDB and the Apache Spark Docker Image, we went to the official Docker Images, but we also used a non-official Docker Image for the mongo-bi-connector. We looked through the Dockerfile that produced the Docker Image looking for anything malicious,

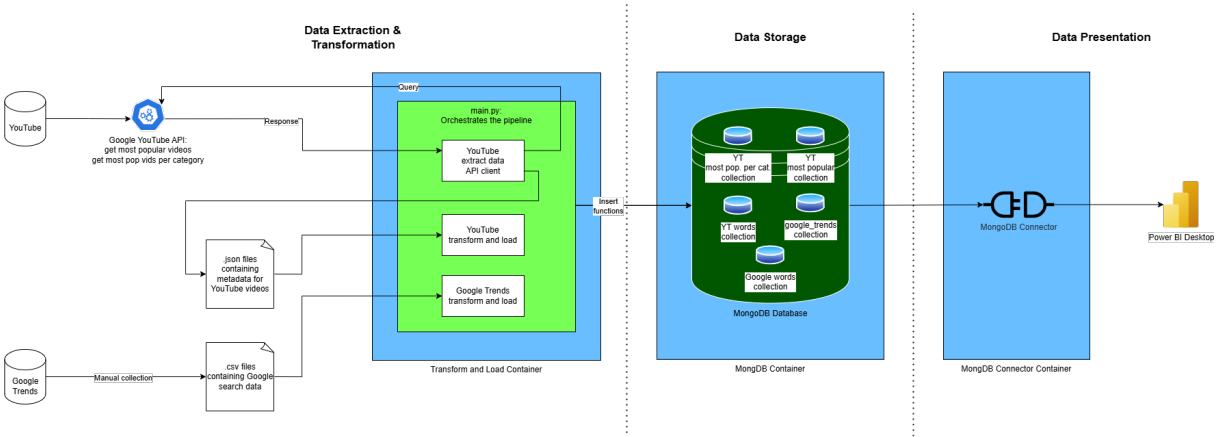

Figure 3.1: System Architecture

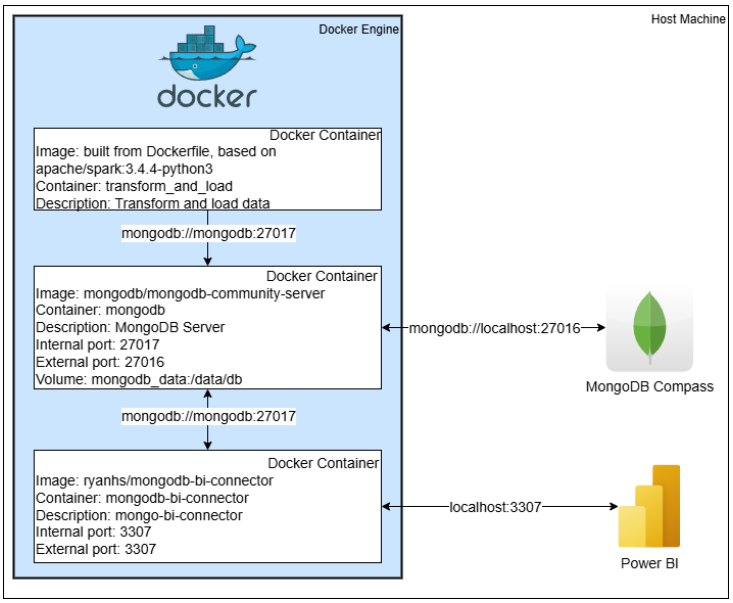

Figure 3.2: Network Diagram

but did not identify any threats.

We created a Dockerfile based on an Apache Spark Docker image and a *docker-compose.yml* file to streamline system deployment. This compose file allowed us to build and start all required services in the form of containers that the system required to function as intended, and it allowed us to do so with one simple docker-compose command.

In this compose file we also created a bridged network that allowed each container that used this network to connect internally. We also exposed the ports for *mongodb* and the *mongodb-bi-connector* containers to the host, so that the host could inspect the data from their host with either the Mongo Shell or with the GUI application, MongoDB Compass. This can be done by connecting to mongodb://localhost:27016.

The default MongoDB ports are in the range 27017 to 27020 [24]. To ensure that the system did not conflict with any existing MongoDB servers, the external port was instead mapped to 27016. Since Microsoft Power BI requires a mongo-bi-connector to connect to the MongoDB server, through the System DSN, we also created a 64-bit MongoDB ODBC 1.4.5 Unicode Driver and connected this data source to the mongodb-bi-connector on port 3307. This meant that all that was required to run the system was to install Docker Desktop which comes with a Docker Engine, we needed to configure an ODBC Data Source for each system, and we needed to install Microsoft Power BI on the host

machine.

The architecture describes a data pipeline that manually collects Google Trends data as CSV files. The ETL container collects data from YouTube Trends and saves this data as JSON files using Google YouTube API. It does this in a delayed loop. After the data has been extracted, a script inside the container reads these files, filters out the required data, and loads the data into the MongoDB container internally on port 27017. The MongoDB Connector Container connects to the database internally on port 27017 and allows Power BI to connect to this connector on port 3307 externally.

3.2 Data Extraction

To deduct societal interests from YouTube (YT) trends, it was necessary to collect relevant data directly from YT. The chosen method for achieving this was through the "Google YouTube API". This API provides access to a wide range of YouTube data, allowing for highly customized queries [25]. To interact with the API, we utilized the Python module "googleapiclient", which enabled us to set up an API client and execute queries to retrieve YT datasets efficiently. The data we deemed most relevant for our analysis was:

- The set of videos included in the "most popular" chart for the region: Norway.
- The set of videos included in the "most popular" chart *per category* for the region: Norway.

These datasets consist of regularly updated lists that provide detailed metadata for each video, including statistics (e.g., view counts and likes), content details (e.g., duration and video definition), and descriptive information (e.g., title, description, and topics) as can be seen in the excerpt in Figure 3.3. For the "most popular" chart, the API supports pagination through a parameter called *nextPageToken*. The amount cap for each query is set to 50 results per page by Google, meaning that we had to collect additional pages of results when more data was available.

For the "most popular" chart we ensured that we got all the data we could get. For category-specific data, we iteratively queried the API for each *videoCategoryId* available

Figure 3.3: Excerpt of the raw data collected from YouTube

in the region (category ID's ranging from 1 to 50). Each query returned a dataset limited to the corresponding category, enabling a fine-grained analysis of trends across different content types. The JSON-ready responses returned by the API were written to local storage as semi-structured JSON files. To maintain clarity and uniqueness, filenames included both a timestamp and an identifier (e.g., page number or category ID). This approach facilitated organization and chronological ordering of the data.

An additional field, *dateTimeAccessed*, was appended to each file to document the exact time the data was retrieved. This ensured traceability and made it easier to analyze trends over time. Error handling mechanisms were implemented to gracefully handle issues such as failed file writes or invalid category queries, ensuring uninterrupted data collection. To manage the API's daily quota limit of 10,000 units, a delay of 216 seconds was introduced between iterations of the data collection process. Each iteration consisted of querying for both the "most popular" chart and the category-specific datasets, with a total cost of 52 quota units per cycle. This allowed for approximately 200 iterations per day, ensuring consistent data collection throughout active hours. This structured approach to data extraction provided a robust foundation for analyzing YT trends in Norway, balancing efficiency, accuracy, and scalability.

The data regarding search results and search trends on google.com can be found on the Google Trends website [26]. For the datasets shown on this website Google does not provide an API. We found an existing python-module named "pytrends" which has

been created for the purpose of extracting data from Google Trends [27]. However, we did not use this in the project because some queries only returned errors and others had unsatisfactory granularity. Despite the absence of an API, Google does provide a

Trends	Search volum	Started	Ended	Trend breakdown	Explore link
man. city mot feyen	5K+	November 26	November 27	man. city mot feyeno	https://trends
sporting lisboa mot	5K+	November 26	November 27	sporting lisboa mot a	https://trends
manchester city	2K+	November 26	November 27	manchester city,city	https://trends
kamzy gunaratnam	2K+	November 26	November 27	kamzy gunaratnam,h	https://trends
man city	1K+	November 26	November 27	man city,man city vs	https://trends
arsenal	1K+	November 26	November 27	arsenal,sporting vs ar	https://trends
evan rachel wood	500+	November 27	November 27	evan rachel wood	https://trends
kvitebjÃ¸rn kong val	500+	November 26	November 27	kvitebjÃ¸rn kong vale	https://trends
fc barcelona â€“ bre	500+	November 26	November 27	fc barcelona â€“ bres	https://trends
bayern mot psg	500+	November 26	November 27	bayern mot psg	https://trends

Figure 3.4: Excerpt from the collected Google Trends search data.

convenient way to acquire the data by providing a download link on the website. These links start the download of the CSV-file containing the data in-view on the website when the desired filters have been applied. Thus, we chose to manually collect the data from the website through the browser in this way. An excerpt of one of the CSV files can be seen in Figure 3.4, showing all the data available to us from the "Trending" page with the following applied filters:

- Location: Norway
- Duration: Last 24h
- Sorting: By search volume descending

The data we were interested in was unfortunately not possible to retrieve historically either from the YT API or from Google Trends (concerning the daily reports), so all data used in this project was both collected and generated during the project's life cycle.

3.3 Data Transformation and Loading

The development phase was divided into three parts as outlined in chapter 5.3. Before data extraction began, all parties agreed on MongoDB as the data storage solution due to its prior success during the lecture week and compatibility with existing Python libraries. To avoid conflicts with existing database servers, we used containers to isolate the results of data collection, considering the risk of colliding databases or collections.

During data extraction, we selected the official *mongodb/mongodb-community-edition* Docker image for its reliability and began configuring it to coexist with existing MongoDB servers. To prevent port conflicts, we exposed port 27016 instead of the default 27017. Additionally, Docker Networking was used to create isolated communication between containers [28]. The exposed port 27016 allowed external connections to the database.

As described in Section 3.2, the first iteration of the Data Extraction process resulted in three files. The data from YouTube Trends was saved in two separate semi-structured JSON files. Whereas Google Trends allowed the extraction of structured CSV-files. Before data transformation could begin, it was necessary to properly understand the structure of these files. Due to the time constraint of the project and the difficulty in automating the process of collecting data from Google Trends, the data from YouTube Trends took the highest priority.

When we use the term "data transformation", we mean filtering, selecting, and manipulating the dataframe which originates from the data from the JSON and CSV files, and thus collecting relevant data for the task at hand, while also creating or transforming new data based on the existing data. We use the term "data loading" for loading the data into the database of choice.

3.3.1 YouTube Trends

The YouTube Trends JSON files contained data on the most popular videos at the time of access, structured into a header, body, and footer. The header described the response type as *videoListResponse*, indicating a list of metadata for trending videos. The body contained a list of items, each providing metadata such as Channel ID, Channel Title, Video ID, Video Title, publication time and date, description, thumbnail metadata, Category ID, and statistics (e.g., views, likes, comments). The footer included details on the number of trending videos and their availability in each file. As noted in Section 3.2, the initial iteration of these files lacked timestamps for data collection, but this was quickly addressed following team feedback.

With the data examined and its structure understood, the next step was data transfor-

mation. During the course lectures and mandatory coursework, we were introduced to Big Data management tools, including Apache Spark and its Python library, *pyspark*. In the design phase, we discussed using Docker Images and containers to streamline system setup, requiring only Docker installation and the Docker Engine, rather than setting up Java 8 and Apache Spark on the host machine. However, since the coursework involved configuring Java 8 and Apache Spark in a Python environment with Conda, and given time constraints to start loading data into the database, the initial solution for data transformation was developed using an existing configured environment in a Jupyter Notebook.

A *SparkSession* was created using *pyspark*. This session is what allows us to perform operations on data sources using Apache Spark. Since the data was stored in the standard file format, JSON, we were able to create a Spark DataFrame by reading the JSON file. A Spark DataFrame is an object with the structure of a table in a database [29]. During the system design phase we decided on using MongoDB with the Python library, *pymongo*, as the interface with the database. We knew that we did not need to normalize the data and that we could store items as documents in a collection in the database.

As soon as pyspark could parse the data, we would start to delve into how pyspark structured the data. With the help of previous exercises, we started with basic functions to show the DataFrame's schema. However, since each video was in a list of items, as previously described, we needed a way to transform each item in the list of items into its entry in a new Spark DataFrame. This can be achieved using Spark's *explode* function [30], by 'exploding' each item in the items list. Once each item was accessible as individual items, we started filtering out the data with a select statement. We had an initial plan for the exact data items we wanted from the JSON file, which was determined before processing the data. The most important data to collect was the video title, the duration of the video, and the amount of views, likes, and comments on the video. However, we also knew that though a video title might change, the video ID would remain immutable. We slowly started filtering out one column at a time. And as we filtered out individual pieces of data, we got a better understanding of what data was available to us.
Another interesting point to add is that since the *_id* field in MongoDB is a unique key for each document in a collection, we had to ensure that we did not add documents more

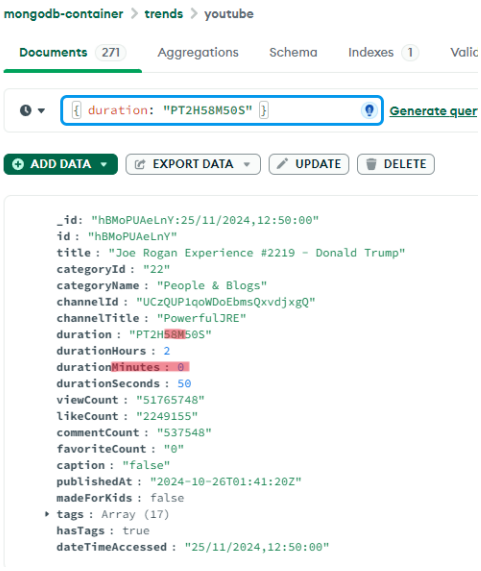

Figure 3.5: Zero Minutes Error

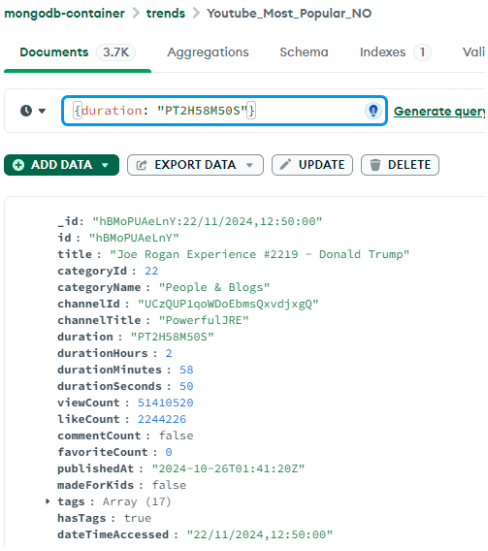

Figure 3.6: Finalized DB Document

than once. Since each JSON file consisted of between 1 to 50 items, we simply made our own format for this unique key. Since the uniqueness of the video ID and the time the data was accessed resulted in creating a form of continuity in the data, we concatenated the *id* with the *dateTimeAccessed*. Each time we started up the system, it would attempt to insert many documents into the collection, but due to this specialized unique key, we could use a *BulkWriteError* exception to catch documents that had already been loaded into the database. This was inspired by a previous solution for an exercise during this course.

After resolving these issues we continued 'selecting' more and more data, with each new piece of data often introducing new challenges. After all the crucial data was in place the visualization of the YouTube Trends data could commence. However, as can be seen in Fig. 3.5, we identified that the data had been loaded as strings, as opposed to integers. Even integer and boolean values were stored as strings, making it difficult to use this data to sort the graphs in Power BI. As soon as this was brought up, we went back to the script and used the *col.cast* function to cast the data to the correct data types which can be seen in Figure 3.6.

3.3.2 Google Trends

A Python script was made to automate the transforming and loading of Google Trends data into our MongoDB database, called "google_trends.py". It processes CSV files containing trend information, standardizes the data, and loads it into a MongoDB collection for efficient storage and retrieval. The following text further explains this script and its components.

The *transform_load* function reads a CSV file using Apache Spark with a predefined schema to ensure consistent data types. It focuses on the *Trend* (the google search term) and *Search vol* columns to simplify the dataset. The file's creation timestamp is extracted and formatted as *dateTimeAccessed* just like we did with the YouTube data. Before inserting data, the MongoDB collection is queried to check if an entry with the same *dateTimeAccessed* already exists, preventing duplicates. The *search_vol* column is cleaned by replacing the letter "K" with "000" and removing any "+"-signs for numeric consistency. The resulting DataFrame is converted into a list of dictionaries compatible with MongoDB's *insert_many* method and is inserted if not empty. This function robustly handles duplicates and ensures data consistency.

The *findTrendFiles* function identifies relevant CSV files in a specified directory by matching a prefix. It returns their absolute paths, ensuring that the correct trend-related files are processed. The *main* function orchestrates the pipeline. It initializes a *SparkSession* for processing, connects to MongoDB to retrieve or create the *google_trends* collection, and defines the schema for Google Trends files using Spark's *StructType* for standardization. It identifies trend files using *findTrendFiles* and processes each file through *transform_load*, ensuring all relevant files are loaded into the database.

The schema enforces data consistency and minimizes ingestion errors. Checking for existing records based on *dateTimeAccessed* avoids redundant storage. Cleaning the *search_vol* column ensures it is ready for numeric computations. File prefix matching facilitates dynamic handling of datasets across regions or time periods. The use of Spark allows scalable processing of large CSV files, while MongoDB's schema-flexible, document-oriented nature suits the dynamic structure of trend data.

3.3.3 YouTube Word Collection

As mentioned earlier, the data collected from YouTube did not always contain tags to identify the themes in each video. This meant that some videos would not contribute to a "word cloud" visual in the visualization step like we envisioned if we only used tags. To solve this problem, the decision was made to try and collect all words from each video's "title", "description" and "tags" fields, clean these words so that they are uniform, remove any "stop words", and then count them up so that a possible trend could be deducted. "Stop words" are the subset of words in a language that can be considered to contain little valuable information in the data collection context [31]. In English, for example, some words that can be said to be stop words are "the", "are" and "to". These stop words were accumulated based on experience and findings (with big help from ChatGPT) in a Python list as can be seen in Figure 3.7.

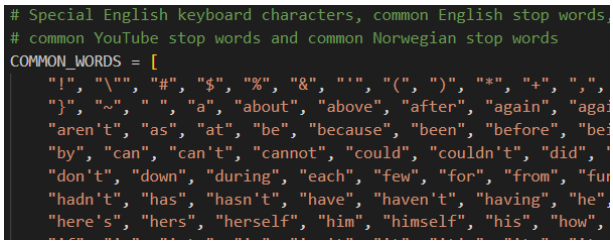

Figure 3.7: Excerpt from the declaration of common stop words and symbols

The script called *yt_words.py* processes our collected YouTube JSON data and loads words contained in it into a MongoDB collection. It uses PySpark to perform key operations including reading JSON files, transforming data to extract and clean word-based features, and then finally loading the results into our database. The *transform_load* function processes a list of JSON files. It reads data into a Spark DataFrame using a predefined schema, extracts the dateTimeAccessed attribute, and checks MongoDB to avoid redundant inserts. Data transformation includes exploding nested arrays, extracting and combining fields (titles, description, tags), de-duplicating words, and filtering out common stop words using a broadcast variable for efficiency. Finally, it groups words by frequency, attaches a date field, and inserts the results into MongoDB.

The *findYtFiles* function scans a directory for JSON files matching a specific prefix and returns their paths. The *findFilesDate* function organizes files by their dateTimeAccessed attribute, creating grouped lists of files sharing the same date. The *main* function initial-

izes a Spark session, establishes a MongoDB connection, and orchestrates the process by iteratively calling the functions described above. The workflow ensures scalability, avoids redundant processing, and efficiently manages large datasets.

3.3.4 Google Trends Word Collection

Just like with the YouTube data, a script was made for the Google Trends data to extract valuable theme-defining words, named "google_words.py". This script processes Google Trends CSV files, extracting key words from trends and their breakdowns, and loads the results into MongoDB. The *transform_load* function processes a single CSV file. It reads the file using a predefined schema, extracts the date from the filename, and verifies if data for that date already exists in MongoDB. If not, it splits the trend and *Trend breakdown* columns into arrays, combines them, removes duplicates, and filters out common stop words using a broadcast variable. The words are then grouped and counted.

A *dateTimeAccessed* field is added to the DataFrame, which is converted to a list of dictionaries and inserted into MongoDB. The *findTrendFiles* function scans a directory for files matching a specified prefix and returns their paths. The main function initializes a Spark session, connects to MongoDB, identifies relevant files, and iterates over them, calling *transform_load* to process and load the data. This approach ensures efficient data handling and prevents redundant processing.

3.3.5 Data Visualization - Power BI

MongoDB Connector for BI

To setup the MongoDB Connector for BI, the MongoDB server was first initialized using "mongod" command inside a command line interface. "mongod" is short for Mongo Daemon and it refers to a MongoDB process running in the background [32]. The BI connector can then be activated be running the executable file "mongosqld" that came included into the download package, specifying the full address of the aforementioned MongoDB server. Starting the database and the mongo-bi-connector are both handled inside of their respective Docker containers, *mongodb* and *mongodb-bi-connector*. For instructions on how to import the database entries to Power BI, please refer to the README.md file.

Extended Data Transformation inside Power BI

While most data processing was performed using Apache Spark, some transformations were done in Power BI. Certain columns in our dataset required data type adjustments to enable effective data visualization:

- The column `dateAccessTime` was stored as a string in MongoDB. To use it as a dependent variable in our dashboard, it was converted to a date/time data type and formatted as *"dd/mm/yyyy, hh:mm:ss"*. The locale was set to "English (Europe)" to ensure consistency.
- Boolean columns, such as `hasTags` and `madeForKids`, were converted to a TRUE/-FALSE data type instead of their original binary representation.

We analyzed the rate of change in YouTube video views over time using a **DAX (Data Analysis Expressions)** [33] query. Although SQL's built-in `PREVIOUSDAY` keyword is typically used for this, gaps in our dataset (due to missing API requests on some days) required an alternative approach. Our query filtered out rows with blank `dateTimeAccessed` cells and identified the most recent valid date before the current date. The view count from this prior date was then retrieved and used to calculate the rate of change. This measure enabled the creation of a bar graph showing `dateAccessTime` and video titles as dependent variables, as illustrated in Figure 3.8.

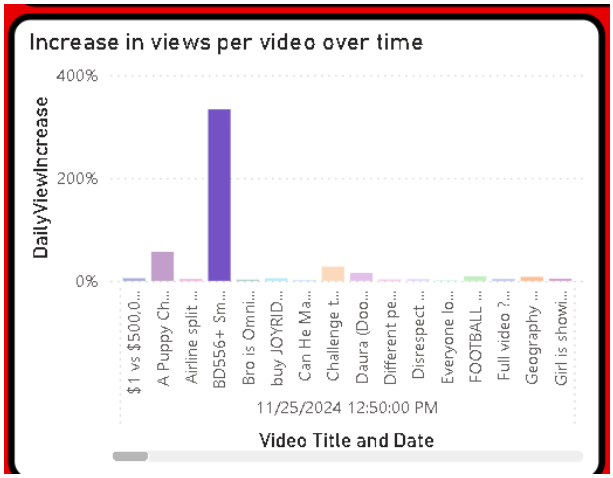

Figure 3.8: Bar graph showing the increase in views of YouTube videos over time

The final queries involved creating two new columns based on three existing ones: `durationHours`, `durationMinutes`, and `durationSeconds`. One DAX query was used to generate the

`VideoTimeFormatted` column, which combines these values into the hh:mm:ss format for easier interpretation of video lengths. However, the hh:mm:ss format is not suitable for data aggregation, which posed a challenge for visuals that analyze the relationship between video durations and factors like view count. To address this, the three columns were converted to seconds and summed to provide the video lengths as a single atomic value. An excerpt of the resulting output is shown in Figure 3.9.

VideoLengthFormatted	VideoLengthInSeconds
03:17:21	*11841*
03:17:21	*11841*
02:58:50	*10730*
02:58:50	*10730*
02:58:50	*10730*
02:58:50	*10730*
02:57:34	*10654*
02:57:34	*10654*

Figure 3.9: Table columns showing the video lengths in hh:mm:ss format and in seconds, respectively

Chapter 4

Results

4.1 Google Trends Data

The CSV files collected from Google Trends was successfully used in our system. The transformation step effectively removed unneeded fields using *pyspark*, and the data frames was thereafter loaded into our project's MongoDB database in their own collection named "google_trends". The decision was made to create one document per search result which then could be ordered using a "dateTimeAccessed" attribute which stated both the date and time the document was collected. An example document from our database showing a Google search term ("trend"), its approximate search volume and the date and time this document was collected can be seen in Figure 4.2. Another example document can be seen in Figure 4.1. It shows a single word collected from the Trends data, how many times it occurred in the file concerning its date, and the date itself.


```
_id: ObjectId('675071d27805869edd3917df')
words : "city"
count : 3
dateTimeAccessed : "27/11/2024"
```


Figure 4.1: Example of Google Trends words document


```
_id: ObjectId('674d9ff951527e3c91e7fc01')
trend : "man. city mot feyenoord"
search_vol : 5000
dateTimeAccessed : "02/12/2024,11:53:19"
```


Figure 4.2: Example of Google Trends document

4.2 YouTube Words Collection

The collection of words from YT was successful, in the sense that it performed as it was designed. The example document in Figure 4.4 shows that useful information can be obtained from these words. In this example, the word "shorts" has been given the number 66, meaning that for all the YT data collected for this day, the word "shorts"

came up in 66 individual videos. That gives us an indication of the popularity of the YT Shorts format. That being said, the solution is not perfect. In Figure 4.3, we can see that the stop word filtering has its limits, taking into account an emoji. Several other stop word/symbols have been included also, which lower the quality of the solution.

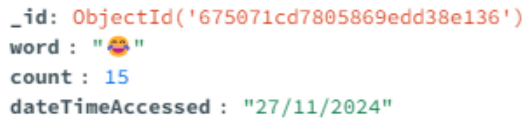

Figure 4.3: Emoji included in the database

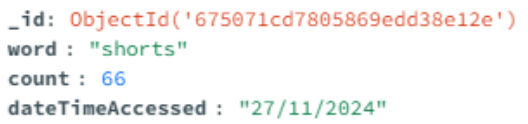

Figure 4.4: One document in the yt_word_count collection

4.3 YouTube Trends Collections

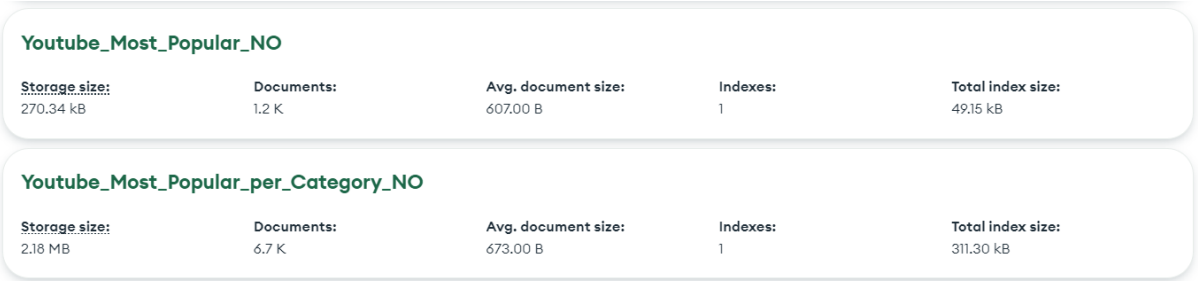

Figure 4.5: YouTube Trends Collections for Norway

The transformation and loading of the YouTube trends JSON files were successful. They have been split into two collections for Norway, YouTube's Most Popular and YouTube's Most Popular per Category. The documents in each collection have the same format, with appropriate datatypes, and can easily be used as an expansion for the Power BI dashboard.

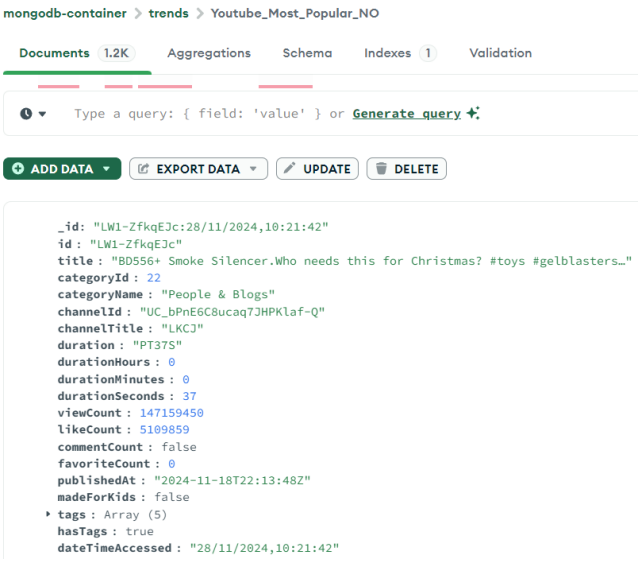

Figure 4.6: Most Popular

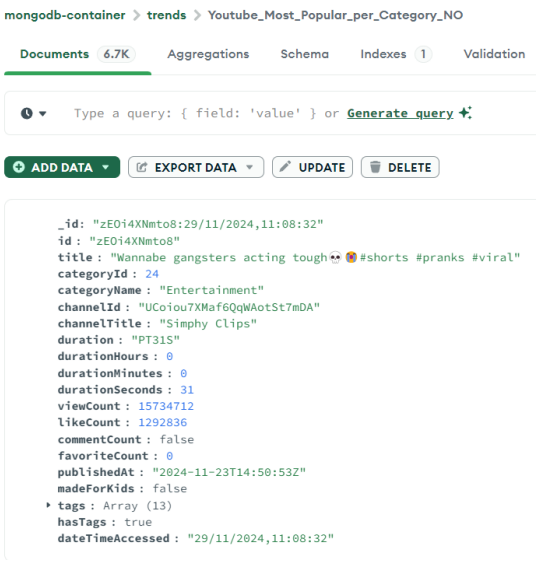

Figure 4.7: Most Popular per Category

4.4 Analyzing Data Using Power BI

4.4.1 Tags and Keywords

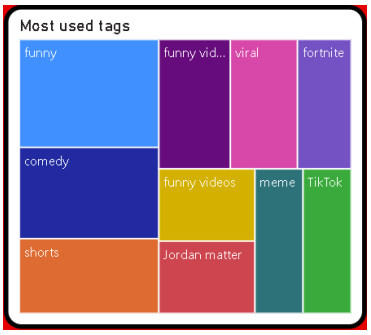

Figure 4.8: Treemap visual showing the most used tags in the YouTube dataset

Analyzing Figure 4.8, the most popular tags were: funny (8), comedy (6), viral (5), shorts (5), funny video (4), viral (3), fortnite (3), meme (3), Jordan matter (3).

4.4.2 Biggest positive change in YouTube views

The top 5 records of videos with the biggest and smallest increase in views can be seen respectively in Table 5.3 and Table 5.4 in the Appendix.
Additional info:

- 4 out 5 videos shown in Table 5.3 are YouTube Shorts. This was confirmed by manually searching the videos on YouTube. The outlier is the video with id aDF

ESN80r8 with is an interview between world-famous footballer Cristiano Ronaldo and Mr. Beast, the most subscribed YouTube channel.

Additional info:

- All of the videos in Table 5.4 are YouTube Shorts.

4.4.3 Video Categories

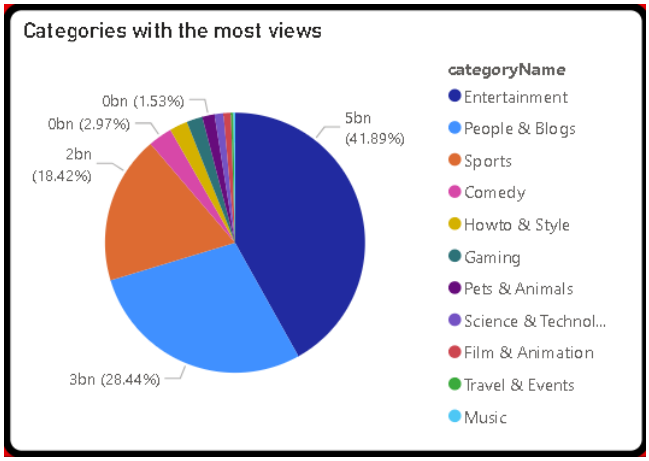

Figure 4.9: Pie chart representing the views that each video category receives.

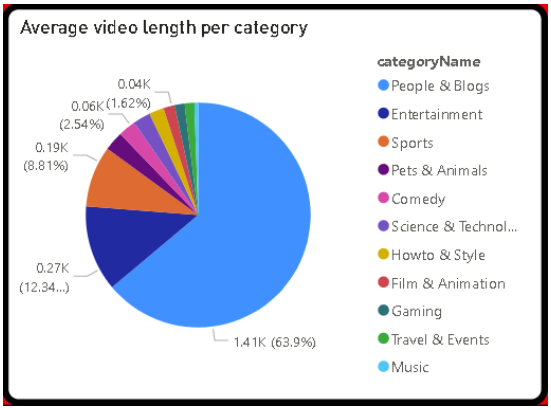

Figure 4.10: Pie chart representing the average length of a YouTube video per video category.

4.4.4 Google Trends

The bar graph in Figure 4.11 showcase the most trending searches on Google on December 4, 2024.

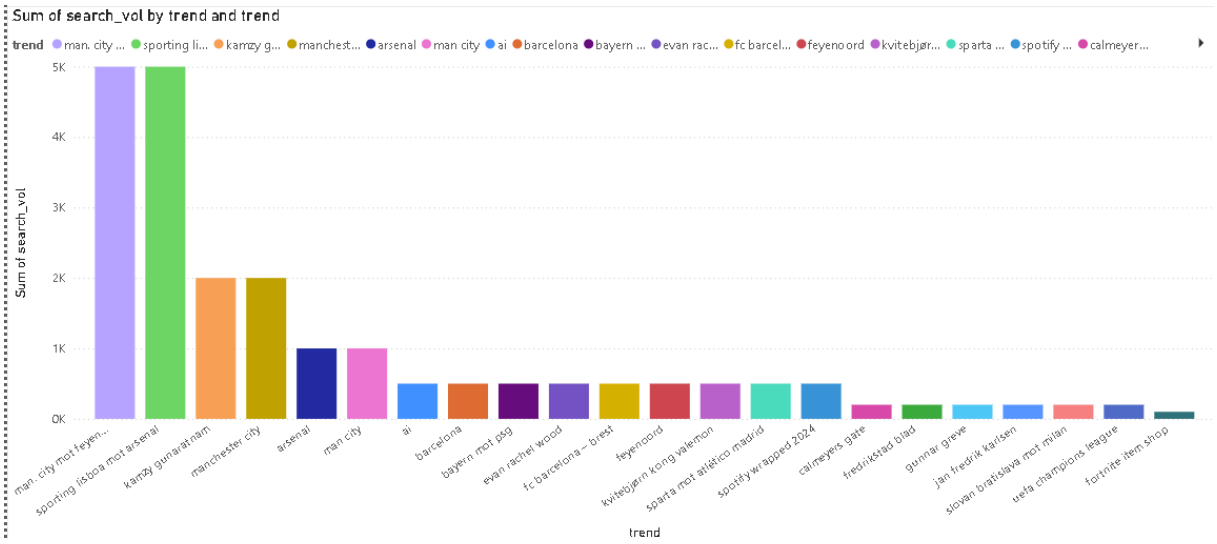

Figure 4.11: Bar chart showing most searched topics on Google in Norway on December 4, 2024

Grouping the results together into fitting categories is done in Table 4.1

Topic	Total
Football	12
Politics	1
Technology	1
Celebrity	3
Art	1
Gaming	1
Other	3

Table 4.1: Figure 4.11 grouped into distinct categories

Chapter 5

Discussion & Conclusion

5.1 Challenges Faced

Although Microsoft Power BI is an effective visualization tool, it lacked certain features we expected, which impacted the usability of some visuals. For example, when hovering over a line in a line graph, the tooltip displays data for all lines, which we found unhelpful. Unfortunately, this is the default behavior and could not be changed. To address this limitation, we included a table in the dashboard that allowed users to search for specific videos, with the option to highlight them. This would then highlight the corresponding line on the graph, improving clarity.

Data collection occurred over a short period (from November 22nd to December 4th, 2024), resulting in a limited variety and volume of data. This smaller dataset made it harder to extrapolate meaningful conclusions with an ideal level of confidence. While this challenge was unavoidable due to time constraints, an earlier start to data collection would have been preferable.

When implementing the transform and load step in our system, we faced an issue with using PySpark in our local software environments. We were not able to run code using PySpark locally at all, and got some cryptic error which we did not succeed in solving. The PySpark package was, however, completely usable in our Anaconda environments with Jupyter Notebook, and thus we developed our Python code there, like in the lectures. After successfully setting up our Docker containers - one with the Spark image - we could move our development from Jupyter Notebook into the "transform_load" container.

For the data transformation and loading of the YouTube Trends data: As soon as we started reading the JSON file issues arose. After consulting the internet, pyspark's documentation, and eventually ChatGPT, we were able to determine that the error was caused due to a missing read option. When pyspark reads a JSON file it expects all the data on one line [34] [35]. However, since the JSON file we were working with consisted of multiple lines we needed to explicitly add a 'multiline' read option.

The video duration was in the ISO-8601 standard duration format which showed difficulty when trying to visualize the individual fields in Power BI. This meant that we could use another pyspark function, *regexp_extract*, which returns a string of the first value that matches the regular expression supplied as the parameter [36]. Again, consulting ChatGPT and explaining which format was used, we were able to create a regular expression that would filter out the duration of seconds and minutes into individual columns. We later realized that we were missing a column for the duration in hours. This helped us to identify an error created with ChatGPT's proposed regular expression, which resulted in zero values in the 'durationMinutes' column for videos that had a duration in hours as shown in Fig. 3.5.

One of the most difficult challenges we faced when transforming the YouTube Trends data, were to handle the tags. This issue arose because YouTube does not create an empty list in the 'snippet' section of the JSON entry when a video lacks tags. Attempting to select a non-existent field using pyspark results in an error. Initially, we implemented logic to check for missing tags in each item and add an empty array when necessary. However, during the final phase of development, we encountered JSON files where the tags section was entirely absent, causing further errors. Again due to the time constraint of the project, and to provide data for the visualization part of the project, we had to consult with ChatGPT in understanding this problem. ChatGPT helped us develop logic to check for the presence of the *item.snippet.tags* field. If the field was missing, the schema was transformed to include an empty array. ChatGPT produced line 102-125 in main.py. While this solution resolved the issue, it significantly impacted the script's performance when handling such cases. Optimizing this process remains an area for improvement.

5.2 Discussion

Analyzing Figures 4.9 and 4.10, the results would indicate that there is indeed a negative correlation between video lengths and the popularity of said videos. In Figure 4.9, it is evident that Entertainment is the most viewed video category by far, with 41.89% of the market share. People & Blogs are at a distant second with 28.44% of the market share. Trailing closer behind is Sports, with 18.42%. The other categories account for similar, but negligible share of the data. However, in Figure 4.10, we see that People & Blogs has on average the longest videos, 1408.3 seconds to be precise, with is approximately 23.5 minutes. Sports and Entertainment however, are considerably shorter on average, 194.22s (ca. 3.5m) and 271.90s (4.5m), respectively. **This would indicate that our hypothesis regarding YouTube video lengths is quite correct**.

However, because there is no featured method for distinguishing YouTube Shorts from other videos other than manually checking, it was not possible to verify this part of our hypothesis: "YouTube Shorts will constitute 30% or more of the platform's most trending videos". Analyzing Figure 4.11, we observe that the majority of Google searches on December 4, 2024 was sports-related. This would at first glance indicate that there is some correlation between Google searches and what is trending on YouTube as Sports made up 18.42% of the video category demographic. However, the majority of the Google searches were related to football fixtures, while YouTube data show that the most popular sports video were about specific football players. The only exception was the video containing highlight from the fixture between Norway and Slovenia in the national men's football ("Slovenia vs. Norge - Sammendrag"), garnering a relatively small amount of 64,338 views. It would also have been preferable to have a larger range of data rather than just December 4th.

The other popular sports Topic on YouTube was the recent boxing match between Jake Paul and Mike Tyson. Observing Figure 5.1, we see that video titled "Jake Paul vs. Mike Tyson FIGHT HIGHLIGHTS ???? — ESPN Ringside" published by ESPN accumulated a staggering 34M+ views since publishing. Analyzing the most rising related entities and queries in Tables Table 5.2 and 5.1, it is clear that this boxing fight was one of the most searched topics on Google in the past 30 days (since December 12th, 2024). Table 5.2 also

shows that Fortnite was a rising topic on Google, which is interesting remembering that Fortnite was a frequently used tag [see Figure 4.8]. All this information could indicate a relationship between Google Trends and YouTube Trends. However, is it difficult based on our results to draw any definitive conclusion about what causes the corresponding change: does the traction on Google searches propagate towards YouTube, or is it the opposite?

A more systematic approach to scheduling sending API requests and downloading Google Trends data could have strengthened our claims by consistently and frequent timestamps. We were unable to calculate any Pearson coefficient between Google Trends and YouTube due to difficulty and time constraints. This makes us unable to answer this part of our hypothesis: "We also expect a positive Pearson correlation ($r > 0.5$) between the societal trends represented in Google Trends and the popularity of videos and categories on YouTube".

Future Work

One idea that we had was implementing a medallion architecture to our system design. The medallion architecture is a design pattern where "with the goal of incrementally and progressively improving the structure and quality of data as it flows through each layer of the architecture" [37]. In our system, this would mean using multiple MongoDB databases, each with their own unique use case. For instance, having a schema that acts as a landing zone for raw data retrieved directly from the original source. And then gradually migrate the collected data to higher tier schemata, each with increasing set of requirements in terms of degree of processing. This idea was abandoned due to time constraints and lack of relevance given our relatively small data set.

The collection and filtration of words from YouTube videos and Google search trends has room for improvement. We suggest both adding many more of and conjugations of stop words in the English/Norwegian languages, to elevate the quality of the filtration. Further, we recommend implementing machine learning to correlate titles, descriptions and tags on YouTube videos with their overall themes to create a completely usable, non-redundant list of keywords for each video.

5.3 Conclusion

Based on our analysis of YouTube and Google Trends data for Norway, several conclusions can be drawn:

- Shorter videos are more popular than longer videos on YouTube. The data indicates a negative correlation between video length and popularity. The "Entertainment" category, featuring shorter videos, holds the largest share of views, while categories with longer videos, like "People & Blogs", have lower viewership. This aligns with the observation that shorter videos, particularly YouTube Shorts, are gaining traction.

- There is a potential correlation between YouTube trends and Google search trends. While the study did not explicitly quantify this correlation, it was designed to explore the relationship between the two platforms. Further investigation is needed to determine the strength and nature of this connection.

The analysis of YouTube metadata, including titles, descriptions, and tags, reveals that YouTube content reflects current societal interests and trends in Norway. Popular tags like "funny," "comedy," "viral," and "shorts" offer insight into what engages audiences. The study encountered some challenges, including:

- Limited data collection period: Data was collected over a relatively short time frame, potentially limiting the variety and volume of data. This could impact the generalizability of the findings.

- Power BI limitations: Certain features in Power BI, such as the inability to customize tooltips for individual lines in a line graph, impacted the usability of some visualizations.

Overall, we believe that this study provides valuable insights into online societal interests in Norway. By leveraging Big Data technologies and a rigorous data analysis approach over data collected from YouTube and Google, we have uncovered meaningful patterns and trends that reflect the evolving interests and behaviors of the online audiences in Norway.

References

[1] *Get started creating YouTube Shorts - YouTube Help*. [Online]. Available: https://support.google.com/youtube/answer/10059070?hl=en (visited on 12/08/2024).

[2] L. Ceci, *YouTube Shorts monthly logged in viewers 2023*, en, Sep. 2023. [Online]. Available: https://www.statista.com/statistics/1314183/youtube-shorts-performance-worldwide/ (visited on 12/08/2024).

[3] G. R. Team, *YouTube Statistics 2024 [Users by Country + Demographics]*, en-US, Dec. 2024. [Online]. Available: https://www.globalmediainsight.com/blog/youtube-users-statistics/ (visited on 12/08/2024).

[4] *List of YouTube Category IDs [2023]*, en-US, Dec. 2022. [Online]. Available: https://mixedanalytics.com/blog/list-of-youtube-video-category-ids/ (visited on 12/08/2024).

[5] V. Nunavath, *Introduction to Big data CS4010*, English, USN, Kongsberg, 2024.

[6] *Understanding How News Works on Google - Google Search*, en. [Online]. Available: https://www.google.com/search/howsearchworks/how-news-works/ (visited on 12/08/2024).

[7] N. A. Rahman, H. J. H. Ng, and V. Rajaratnam, "Big data analysis of a dedicated youtube channel as an open educational resource in hand surgery," *Frontiers in Applied Mathematics and Statistics*, vol. 7, 2021, ISSN: 2297-4687. DOI: 10.3389/fams.2021.593205. [Online]. Available: https://www.frontiersin.org/journals/applied-mathematics-and-statistics/articles/10.3389/fams.2021.593205.

[8] *Mayer's 12 Principles of Multimedia Learning — DLI*, en. [Online]. Available: https://www.digitallearninginstitute.com/blog/mayers-principles-multimedia-learning (visited on 12/08/2024).

[9] *Overview - Spark 3.5.3 Documentation*, [Online; accessed 6. Dec. 2024], Sep. 2024. [Online]. Available: https://spark.apache.org/docs/latest/#apache-spark-a-unified-engine-for-large-scale-data-analytics.

[10] *Apache spark™ - unified engine for large-scale data analytics*. [Online]. Available: https://spark.apache.org/.

[11] *Introducing json*. [Online]. Available: https://www.json.org/json-en.html.

[12] *Iso - iso 8601 — date and time format*, en, Feb. 2017. [Online]. Available: https://www.iso.org/iso-8601-date-and-time-format.html.

[13] *Iso 8601 duration format*, en. [Online]. Available: https://www.digi.com/resources/documentation/digidocs/90001488-13/reference/r_iso_8601_duration_format.htm.

[14] davidiseminger, *What is power bi desktop? - power bi*, en-us, Jan. 2024. [Online]. Available: https://learn.microsoft.com/en-us/power-bi/fundamentals/desktop-what-is-desktop.

[15] Ibm, *What Is MongoDB?* [Online; accessed 6. Dec. 2024], Aug. 2024. [Online]. Available: https://www.ibm.com/topics/mongodb.

[16] *What is the MongoDB Connector for BI? - BI Connector*, en. [Online]. Available: https://www.mongodb.com/docs/bi-connector/current/what-is-the-bi-connector/ (visited on 12/08/2024).

[17] *Docker docs - sharing local files with containers*, en. [Online]. Available: https://docs.docker.com/get-started/docker-concepts/running-containers/sharing-local-files/.

[18] *Docker engine security*, en, 2024. [Online]. Available: https://docs.docker.com/engine/security/.

[19] *PySpark Overview — PySpark 3.5.3 documentation*, [Online; accessed 6. Dec. 2024], Sep. 2024. [Online]. Available: https://spark.apache.org/docs/latest/api/python/index.html.

[20] *PyMongo 4.10.1 documentation*, [Online; accessed 6. Dec. 2024], Oct. 2024. [Online]. Available: https://pymongo.readthedocs.io/en/stable.

[21] *Get Started with PyMongo - PyMongo v4.10*, [Online; accessed 6. Dec. 2024], Dec. 2024. [Online]. Available: https://www.mongodb.com/docs/languages/python/pymongo-driver/current/get-started.

[22] *Google API Client Library for Python Docs*, [Online; accessed 6. Dec. 2024], Dec. 2024. [Online]. Available: https://googleapis.github.io/google-api-python-client/docs.

[23] *google-api-python-client*, [Online; accessed 6. Dec. 2024], Dec. 2024. [Online]. Available: https://github.com/googleapis/google-api-python-client.

[24] *Default mongodb port - mongodb manual v8.0*, en. [Online]. Available: https://www.mongodb.com/docs/current/reference/default-mongodb-port/.

[25] *API Reference | YouTube Data API | Google for Developers*, [Online; accessed 26. Nov. 2024], Dec. 2023. [Online]. Available: https://developers.google.com/youtube/v3/docs.

[26] *Google Trends*, [Online; accessed 26. Nov. 2024], Nov. 2024. [Online]. Available: https://trends.google.com/trends.

[27] *pytrends*, [Online; accessed 26. Nov. 2024], Nov. 2024. [Online]. Available: https://pypi.org/project/pytrends.

[28] *Docker networking overview*, en, 2024. [Online]. Available: https://docs.docker.com/engine/network/.

[29] *R on spark - spark 3.5.3 documentation*. [Online]. Available: https://spark.apache.org/docs/3.5.3/sparkr.html#sparkdataframe.

[30] *Sql, built-in functions, explode*. [Online]. Available: https://spark.apache.org/docs/latest/api/sql/index.html#explode.

[31] Wikipedia contributors, *Stop word — Wikipedia, the free encyclopedia*, [Online; accessed 5-December-2024], 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Stop_word&oldid=1257866038.

[32] *Quick Start Guide for Windows - BI Connector*, en. [Online]. Available: https://www.mongodb.com/docs/bi-connector/current/local-quickstart/ (visited on 12/08/2024).

[33] kfollis, *DAX Queries - DAX*, en-us, Oct. 2023. [Online]. Available: https://learn.microsoft.com/en-us/dax/dax-queries (visited on 12/08/2024).

[34] N. Nelamali, *Pyspark read multiple lines (multiline) json file*, en-US, Dec. 2019. [Online]. Available: `https://sparkbyexamples.com/pyspark/pyspark-read-multiple-line-json-file/`.

[35] N. Nelamali, *Pyspark read json file into dataframe*, en-US, Dec. 2020. [Online]. Available: `https://sparkbyexamples.com/pyspark/pyspark-read-json-file-into-dataframe/`.

[36] *Sql, built-in functions, regexp_extract*. [Online]. Available: `https://spark.apache.org/docs/latest/api/sql/index.html#regexp_extract`.

[37] *What is a Medallion Architecture?* en-US, Mar. 2022. [Online]. Available: `https://www.databricks.com/glossary/medallion-architecture` (visited on 12/08/2024).

Appendices

Search Term	**Search Growth (%)**
Mattelabbens mester	2350%
Mattelab	1600%
Mike Tyson vs Jake Paul	1000%
Håndball EM 2024	1000%
Spotify Wrapped 2024	950%
Mike Tyson	900%
Maskorama	850%
XRP	800%
Olav Thon	650%
XRP Price	650%
Bodø Glimt Manchester United	350%
Dogecoin	350%
Firi	350%
Maskorama 2024	350%
Gladiator 2	350%
Juletre	350%
Gladiator	350%
Wicked	350%
Black Friday	250%
Ønskeskyen	190%
Yellowstone	170%
Bitcoin Price	170%
Bitcoin	170%
Sabrina Carpenter	160%
Bitcoin Kurs	140%

Table 5.1: Google Trends - Trending Queries the past 30 days (since december 12, 2024)

Search Term	**Search Growth (%)**
Jake Paul	1200%
Mike Tyson	1000%
Maskorama	850%
Christmas tree	350%
Black Friday	300%
Christmas	170%
December	170%
Bitcoin	170%
Prisjakt Norge NUF	130%
Handball	110%
Spotify	80%
Gift	80%
Advent calendar	80%
Elkjøp	70%
The LEGO Store	60%
XXL	60%
PlayStation 5	60%
Fortnite	60%

Table 5.2: Google Trends - Trending Topics the past 30 days (since December 12, 2024)

id	title	access date	duration	total views	increase in views
06kY6gqHj-I	The bench is freshly painted ???? with @SofiManassyan @CarterKench #Comedy #Friends #andragogan	11/25/2024 12:50:00 PM	00:00:13	10,380,143	363.61%
LW1-ZfkqEJc	BD556+ Smoke Silencer. Who needs this for Christmas? #toys #gelblasters #gelblasterguns #airsoft	11/25/2024 12:50:00 PM	00:00:37	116,182,276	333.46%
vrvRJWak0nw	Why no RONALDO?! ????	11/25/2024 12:50:00 PM	00:00:28	70,439,783	170.57%
aDF_ESN80r8	I Meet MrBeast To Break The Internet!!	11/25/2024 12:50:00 PM	00:14:00	41,081,896	137.90%
aPrAI70J0AQ	Why is there corn? #chillguy #motivation	11/27/2024 12:36:27 PM	00:00:23	5,138,495	85.59%

Table 5.3: Videos with the biggest views increase

id	title	access date	duration	total views	increase in views
uLqoe3NvZQc	When Freddie Freeman found a kid dressed up as him for Halloween ???????? (via chelseafreeman/TT) #shorts	11/26/2024 10:08:35 AM	00:00:12	11,298,982	0.09%
Q3woJb3ZxnA	Motorbike Smashes Into Porsche! ????	11/26/2024 10:08:35 AM	00:00:15	23,461,683	0.10%
uLqoe3NvZQc	When Freddie Freeman found a kid dressed up as him for Halloween ???????? (via chelseafreeman/TT) #shorts	11/27/2024 12:36:27 PM	00:00:12	11,314,683	0.14%
OLbNpydPrOI	Everyone loves high fives at the leaning tower of Pisa ???????????? ????: jordybutler_ via BViral	11/26/2024 10:08:35 AM	00:00:16	5,383,403	0.15%
I9b-7KpL6e8	Mike Tyson Reveals Why He Won Against Jake Paul	11/26/2024 10:08:35 AM	00:00:25	10,248,329	0.16%

Table 5.4: Videos with the smallest views increase

title	Accessed	Length	# of Views	Increase in views
Players vs Pitch ????	11/27/2024 12:36:27 PM	00:00:26	137223507	0.67%
Players vs Pitch ????	11/26/2024 10:08:35	00:00:26	136306224	0.71%
Players vs Pitch ????	11/25/2024 12:50:00 PM	00:00:26	135347523	3.64%
Players vs Pitch ????	11/22/2024 12:50:00 PM	00:00:26	130589859	
Why no RONALDO?! ????	11/27/2024 12:36:27 PM	00:00:28	83552874	8.13%
Why no RONALDO?! ????	11/26/2024 10:08:34	00:00:28	77267223	9.69%
Why no RONALDO?! ????	11/25/2024 12:50:00 PM	00:00:28	70439783	170.57%
I Meet MrBeast To Break The Internet!!	11/27/2024 12:36:27 PM	00:14:00	44640349	3.65%
Disrespect or Respect ????	11/27/2024 12:36:27 PM	00:00:27	43675921	0.64%
Disrespect or Respect ????	11/26/2024 10:08:35	00:00:27	43400246	0.52%
Disrespect or Respect ????	11/25/2024 12:50:00 PM	00:00:27	43177739	3.63%
I Meet MrBeast To Break The Internet!!	11/26/2024 10:08:34	00:14:00	43068648	4.84%
Disrespect or Respect ????	11/22/2024 12:50:00 PM	00:00:27	41665459	
I Meet MrBeast To Break The Internet!!	11/25/2024 12:50:00 PM	00:14:00	41081896	137.90%
Jake Paul vs. Mike Tyson FIGHT HIGHLIGHTS ???? | ESPN Ringside	11/27/2024 12:36:27 PM	00:02:35	34049577	0.34%

Figure 5.1: Most watched sports videos in our dataset

Acknowledgements

NOTE: All contributing members used ChatGPT 4o for grammar and spell checking, and for writing and troubleshooting code.

During the project proposal phase, we explored previous datasets from Kaggle but decided to create our own dataset from a reliable source to better understand the data generation process. During the lecture week, we were introduced to the ETL (Extract, Transform, Load) pipeline and data visualization concepts, which inspired us to divide the project into three parts, aligning with the group size. This division resulted in a data pipeline with distinct responsibilities for each group member.

Candidate 8552 handled data extraction and storage in standardized formats (JSON and CSV). Candidate 8560 was responsible for transforming and loading this data into a MongoDB database. Candidate 8581 focused on visualizing and analyzing the data using Microsoft Power BI.

We established clear interfaces between the pipeline stages: standardized file formats for Spark compatibility between extraction and transformation, *pyspark* and *pymongo* for transformation and loading, and the MongoDB BI Connector for visualization. These interfaces ensured parallel work could proceed efficiently, though latency between stages required effective coordination to prevent bottlenecks.

After allocating responsibilities, we collaboratively developed a system architecture diagram, which helped us align on the project's structure and goals. Leveraging prior experience with Docker and Docker Compose, we opted for a multi-container approach, treating each component as a service. The final architecture, refined through minor revisions, is shown in Fig.3.1.

Here we declare the individual contributions to the project both in the implementation and in writing the report.

Candidate No.: 8552

Work done:

- Extraction of all data used in this project with the YT API and manual collection of Google Trends files
- Transforming and loading of the Google Trends files. Word extraction, filtering and ordering of YouTube and Google data.
- Implementation of the following scripts:
  - google_trends.py
  - google_words.py
  - yt_getMostPopular.py
  - yt_words.py
  - Contributions in supporting scripts
- Main responsibility for the overall system architecture diagram

Contributions in the report include the following sections:

- Method/Data Extraction - section 3.2
- Method/Data Transformation and Loading/Google Trends - subsection 3.3.2
- Method/Data Transformation and Loading/YouTube Word Collection - subsection 3.3.3
- Method/Data Transformation and Loading/Google Trends Word Collection - subsection 3.3.4
- Results/Google Trends Data - section 4.1
- Results/YouTube Words Collection - section 4.2

- Theoretical background/MongoDB - section 2.2
- Theoretical background/Python Packages - section 2.2
- Contributions in the following sections:
  - Challenges Faced section 5.1
  - Conclusion section 5.3
  - Discussion section 5.2
  - Future Work section 5.2

Candidate No.: 8560

Work done:

- Setting up database, system requirements, and system as containers
- Transforming and loading the YouTube Trends files
- Implementation of the following:
  - pyspark_noteboook.ipynb
  - Dockerfile
  - docker-compose.yml
  - main.py
- Main responsibility for the system integration

Contributions in the report include the following sections:

- Introduction/Problem Statement - section 1.2
- Theoretical Background/Technologies and Standards - section 2.2
- Method/System Architecture - section 3.1
- Method/Data Transformation and Loading - section 3.3
- Method/Data Transformation and Loading/YouTube Trends - subsection 3.3.1

- Results/YouTube Trends Collections - section 4.3
- Discussion & Conclusion/Challenges Faced - section 5.1

Candidate No.: 8581

Work done:

- All work related to data visualization in Microsoft Power BI, including the creating of all dashboards and DAX.
- Analyze the results from the processed data from YouTube and Google trends using Power BI.
- Conducted and written the literature review 2.1
- Researched, set up and written about MongoDB Connector for BI 2.2
- Suggested idea of using medallion architecture for future work 5.2.

Contributions in the report include the following sections:

- Introduction (Introduction, Motivation, Problem Statement and Hypothesis) chapter 1
- Literature Review section 3.2
- Technologies and Standards (MongoDB Connector for BI) section 2.2
- Results (Analyzing Data Using Power BI) section 4.4
- Discussion & Conclusion (Challenges Faced, Discussion, Future Work) section 5.1 section 5.2 section 5.2