IT350 Assignment 3 - Report

Sachin Prasanna - 211T058

March 6, 2024

1 Problem Statement

Use the allocated data source (any resource given below) and build a utility to extract and curate data for the analytics tasks (Eg. Hate Speech Detection, etc) specified. Mention any keywords used for search, Ensure that this module provides a Live Stream of data. – (3 marks)

Using the data collected as part of question 1, perform the given task, without using bloom filter with the data stream and with the bloom filter applied to the data stream (Use an existing library if available for this part) – (2+2 Marks)

Finally, build a visualization module for data obtained from the task carried out in part 2 so that any changes in the stream are reflected in the visualization. Visualization will be done using time series plot, heat map and word clouds for both before and after application of bloom filter. Also compute execution time before and after applying bloom filter and give your reasons for the results obtained– (3 Marks).

Task Allocated according to my roll number : Topic Identification of Tumblr posts

2 API Configurations

For connecting and retrieving posts from Tumblr, the Pytumblr library was used. Pytumblr is a Python library for interacting with the Tumblr API, simplifying data access and enabling various tasks programmatically.

The TumblrRestClient class facilitates requests to Tumblr's RESTful API, serving as the primary interface. Users authenticate with credentials for secure access to Tumblr data. This enables development of tools for content discovery, analytics, and social media management.

3 Topics for Identification

For this assignment, I have taken three main topics which are to be identified from the stream of posts. They are **Programming**, **Cooking** and **Sports**. All other posts are categorised as **Others**. Keywords are specified for each topic and based on these keywords, posts are pulled from the Tumblr website. I have taken 30 keywords for each topic.

4 Bloom Filters

Bloom filters are space-efficient probabilistic data structures used for testing whether an element is a member of a set. They offer fast membership queries and require minimal memory compared to other data structures like hash tables or trees.

Bloom filters employ a fixed-size bit array along with multiple hash functions. Upon adding an element, it undergoes hashing by these functions, setting respective bits in the array to 1. For membership verification, the element is hashed again using the same functions. If all corresponding bits are set to 1, it's likely within the set; otherwise, it's conclusively not.

The Bloom Filter was implemented using the mmh3 library for generating hash functions and the bitarray library for fast creation of bitarrays. A class based implementation is done further and options to add and contains are specified accordingly.

5 Dataset Preparation

The dataset for Topic Identification was obtained and constructed through Tumblr API calls. Initially, 30 keywords for the topics **Programming**, **Cooking**, **Sports**, and **Other** were specified, closely related to their respective topics.

API calls were then sent to retrieve 20 posts for each keyword. Thus, each topic will have a maximum of $20 \times 30 = 600$ posts, assuming all keywords yield 20 posts each. The attributes in the dataset include:

- **Post Timestamp**: Time at which the post was created.
- **Post ID**: Unique identifier for each Tumblr post.
- **Blog Name**: Name of the Tumblr blog from which the post originated.
- **Post Type**: Type of post (e.g., text, photo, video).
- **Tags**: Tags associated with the post, providing insight into its content.
- **Post Summary**: A brief summary of the post content, if available.
- **Label**: The assigned label for the post, indicating its topic (e.g., Programming (0), Cooking(1), Sports(2), Other(3)).

6 Preprocessing the Dataset

The data preprocessing stage is a crucial step in preparing text data for analysis. The first step of preprocessing was iterating over each row in the DataFrame, appending the tags from the 'Tags' column to the end of the 'Post Summary' column, and updating the 'Post Summary' column with the modified text. Thereafter, A series of preprocessing techniques have been applied to clean the text data stored in the 'Post Summary' column of the DataFrame. They include:

- **Lowercasing:** All text values are converted to lowercase to ensure consistency in processing.
- **URL Removal:** URLs are commonly present in text data and are often irrelevant to analysis. Therefore, a regular expression pattern is used to remove URLs from the text.
- **Punctuation Removal:** Punctuation marks are stripped from the text to focus on meaningful words.
- **Stopword Removal:** Stopwords, which are common words that do not carry significant meaning, are removed from the text. This helps in reducing noise in the data and improves the quality of analysis.
- **Lemmatization:** Lemmatization is performed to reduce words to their base or root form. This process helps in standardizing words and reducing the vocabulary size.

7 Machine Learning for Topic Identification

The **Linear Support Vector Machine** (SVM) model developed in this project aims to classify textual data into the different topics. The model is trained on a dataset consisting of posts from various sources, each labeled with a specific topic such as programming, cooking, sports, or other.

7.1 Model Development

Data Preprocessing: The dataset is preprocessed to handle missing values by dropping rows with missing post summaries. Text data is then vectorized using a CountVectorizer to convert text into numerical features and a TfidfTransformer to transform them into TF-IDF (Term Frequency-Inverse Document Frequency) representation, which captures the importance of words in the context of the entire corpus.

Model Training: The SVM model is trained using the Stochastic Gradient Descent (SGD) classifier with a hinge loss function. The classifier is configured with hyperparameters such as alpha (regularization parameter), maximum number of iterations, and random state for reproducibility. The training data is split into training and testing sets using a standard 80-20 split.

Model Evaluation: The performance of the trained model is evaluated using accuracy, which measures the proportion of correctly classified instances in the test set. The accuracy score provides insights into how well the model generalizes to unseen data. The model performed extremely well and had an accuracy **95.14%** for the validation set.

Model Serialization: Once trained, the model is serialized using the Joblib library, allowing for easy storage and reuse without the need for retraining.

Prediction Functionality: The developed model includes a prediction function that takes a new sentence as input and predicts its corresponding topic label. The function loads

the serialized model, predicts the label for the input sentence, and returns the predicted topic category based on predefined mappings.

8 Live Data Stream

The process of fetching and filtering Tumblr posts based on specified keywords, with a primary focus on comparing the efficiency of data retrieval with and without the application of a Bloom filter is implemented here.

Initially, additional tags related to programming, cooking, sports, and other topics are defined, along with a list containing all keywords. These keywords are then shuffled to ensure randomness in the selection process. A Bloom filter is then initialized with a specified size and number of hash functions, and it is populated with all keywords associated with programming, cooking and sports. The main point here is the Bloom Filter will only react positively to those posts which are related to **Sports**, **Programming** or **Cooking**.

Subsequently, Tumblr API calls are made to retrieve posts based on the shuffled keywords, with and without the Bloom filter. One function retrieves posts while utilizing the Bloom filter to filter out irrelevant posts (posts which are not either one of the 3 topics. For each retrieved post, the function checks if any of its tags match the keywords stored in the Bloom filter. If a match is found, the post data is stored for further processing.

Similarly, the another function retrieves posts without employing a Bloom filter. All retrieved posts are processed without any filtering based on keywords.

Finally, the execution time with and without the Bloom filter is recorded for comparison. This comparison provides insights into the efficiency gain achieved by utilizing a Bloom filter for post filtering. The implementation highlights the effectiveness of using a Bloom filter for efficient post filtering, reducing processing time and improving overall efficiency in data retrieval, particularly in large-scale data processing tasks.

It was observed that using Bloom Filters are indeed beneficial, as it took only **45 seconds** to retrieve relevant topic posts with the bloom filter, as opposed to **52 seconds** without the Bloom Filter. This lesser time taken of Bloom Filters can mainly attributed to the following 2 reasons:

- **Reduced Data Retrieval Time:** Bloom Filters efficiently filter out other posts (which are not any of the 3 topics), reducing the amount of data that needs to be processed and retrieved from the source. This significantly decreases the time required for data retrieval, resulting in faster processing and response times.
- **Optimized Resource Utilization:** With Bloom Filters, the system can allocate its resources more efficiently by focusing only on relevant data. This optimization ensures that computational resources are not wasted on processing posts related to Other topics, leading to faster execution times.

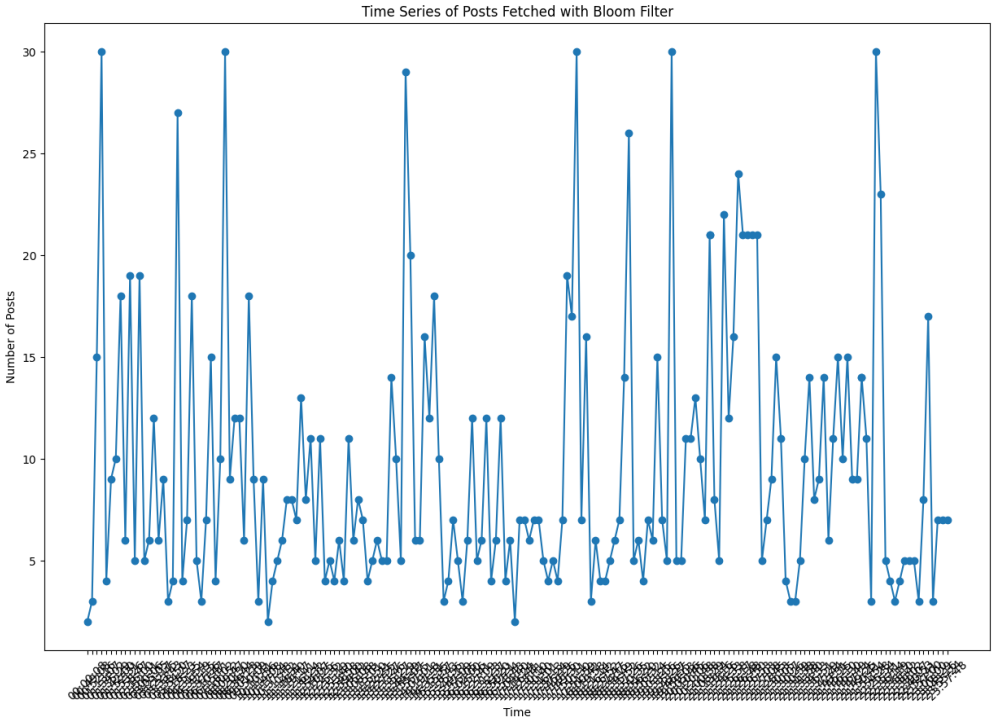

Figure 1: Time Series of Posts Fetched with Bloom Filter

9 Data Visualisation and Results

9.1 Time Series Plots

The results on the Time Series Plots are shown from Figure 1 to Figure 6. It is clear that the graphs using the Bloom Filters have lesser posts processes as compared to the graphs without using the Bloom Filters. This is the advantage of Bloom Filters as it filters out posts using keywords given.

9.2 WordCloud

The Worldcloud is constructed by taking in the tags and the summary of the posts. The results of the WordCloud are shown from Figure 7 to Figure 14. Both without application of Bloom Filter and application of Bloom Filter yields similar results for the 3 chosen categories (Programming, Cooking, Sports), which shows that even though the Bloom Filter filters out posts, it does not lose a lot of information of the incoming data stream.

9.3 Heat Maps

The heat map is used to generate a heatmap visualization representing the frequency of different tags over time. This visualization allows for the observation of trends and patterns in tag frequency changes over time. Figure 15 and Figure 16 capture the results. It is

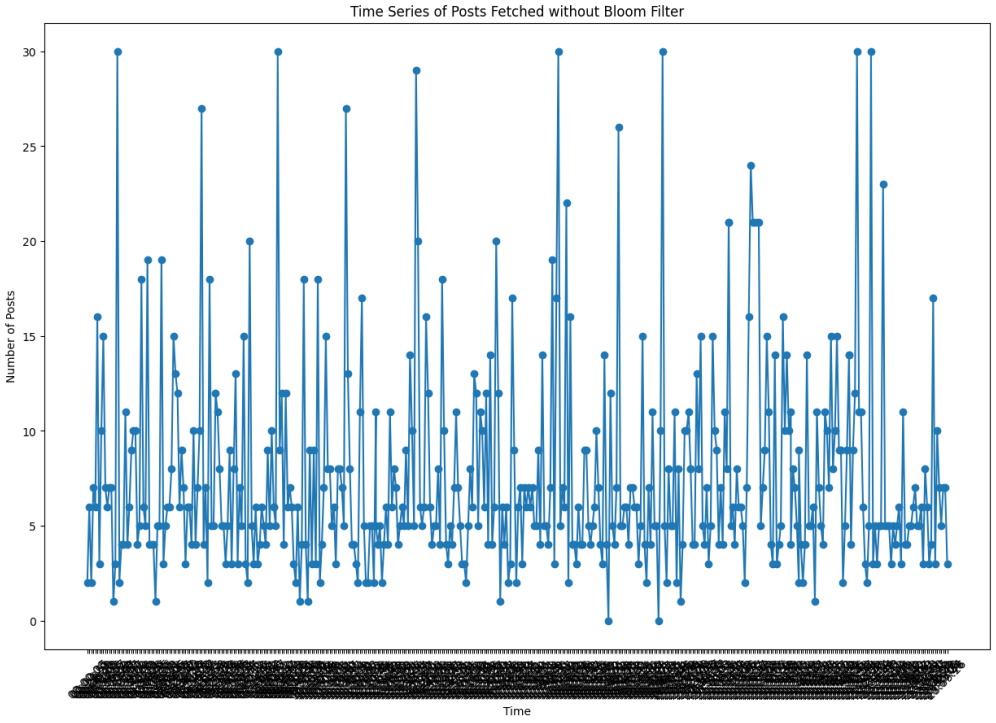

Figure 2: Time Series of Posts Fetched without Bloom Filter

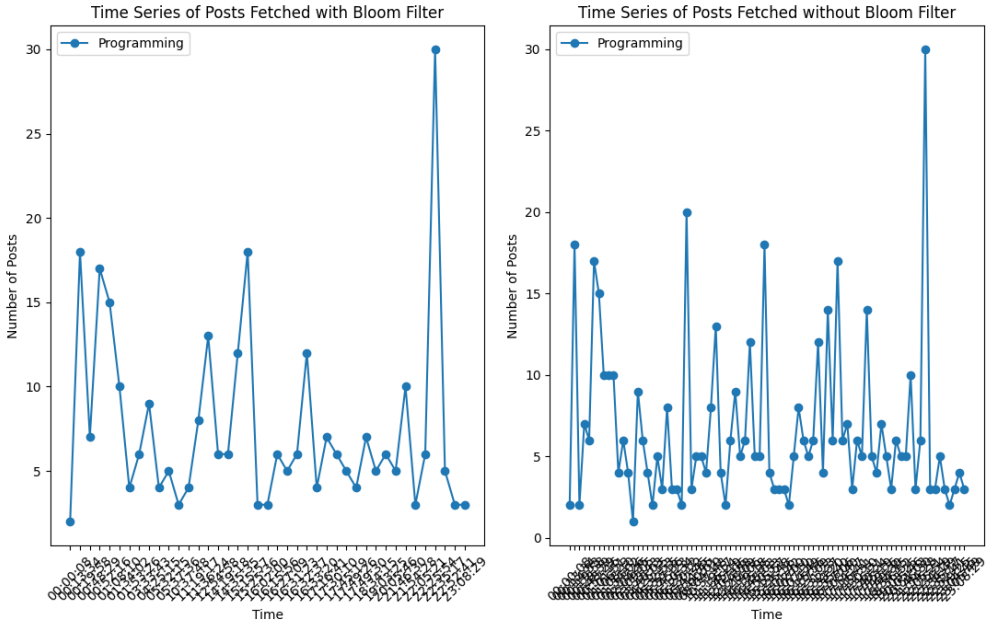

Figure 3: Comparison of Programming Posts fetched with Bloom Filter and without Bloom Filter using Time Series

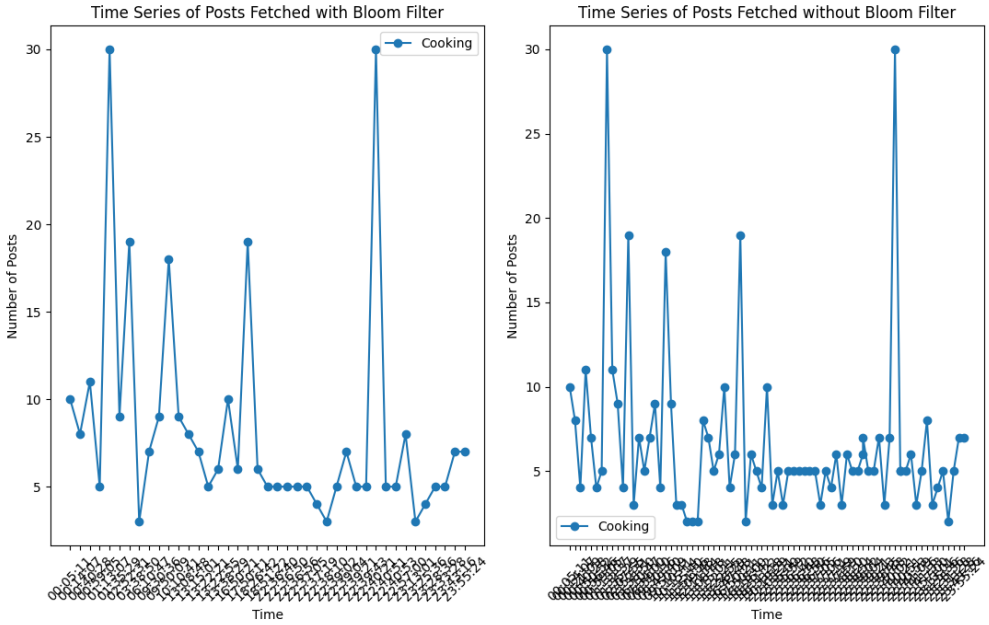

Figure 4: Comparison of Cooking Posts fetched with Bloom Filter and without Bloom Filter using Time Series

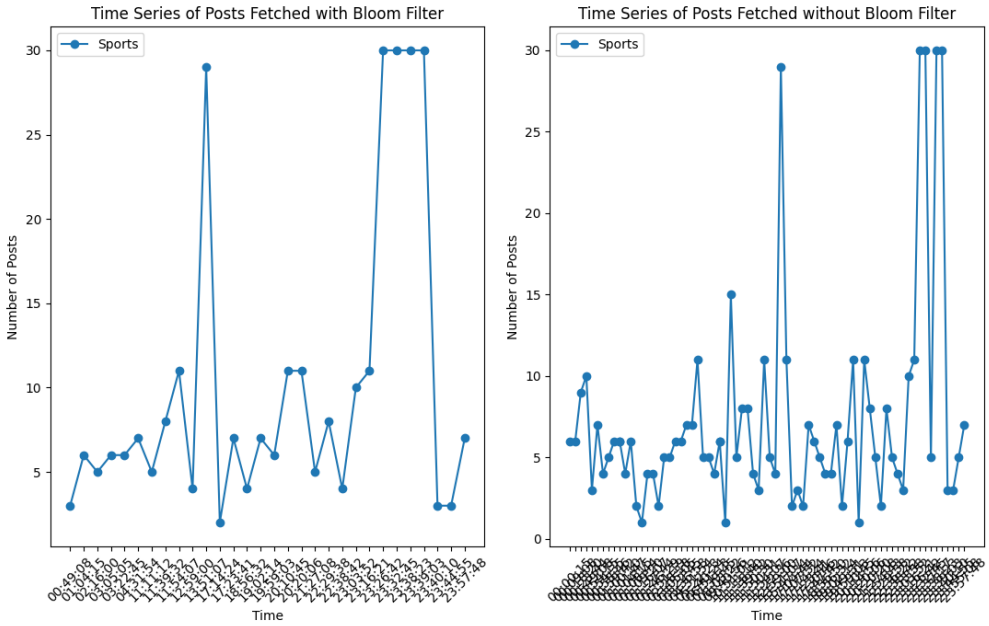

Figure 5: Comparison of Sports Posts fetched with Bloom Filter and without Bloom Filter using Time Series

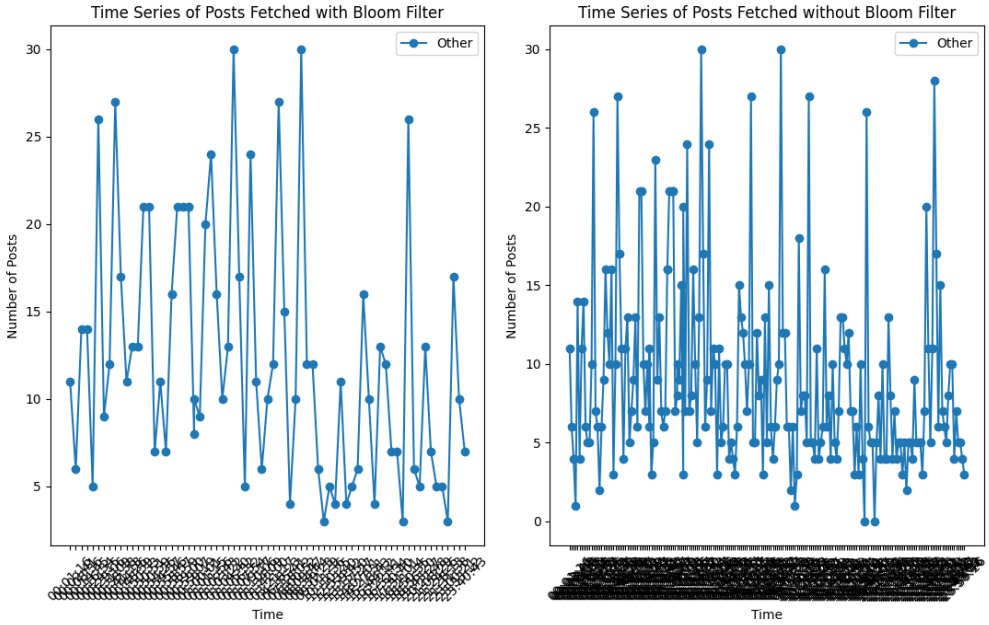

Figure 6: Comparison of Other Posts fetched with Bloom Filter and without Bloom Filter using Time Series

Figure 7: World Cloud of Programming Posts after applying Bloom Filters

Figure 8: World Cloud of Programming Posts before applying Bloom Filters

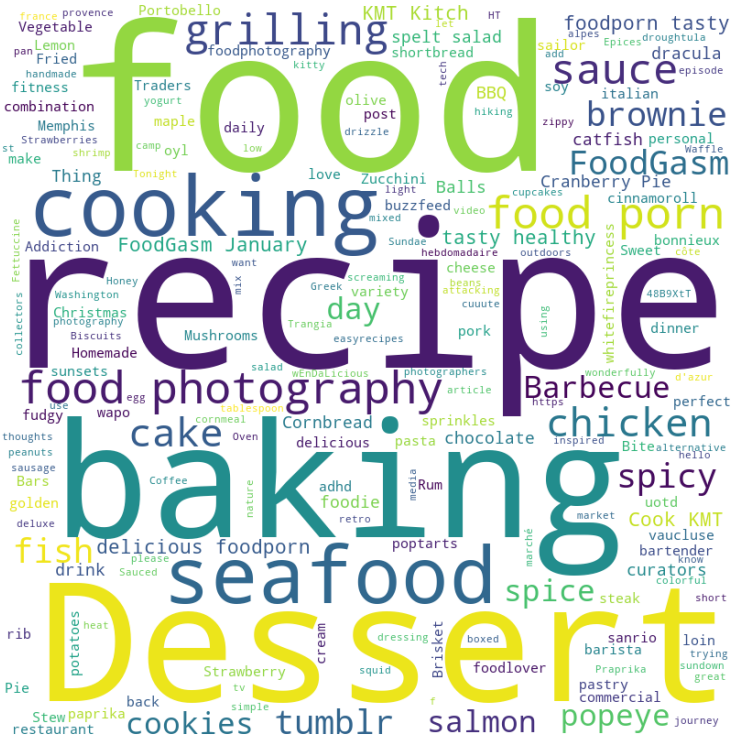

Figure 9: World Cloud of Cooking Posts after applying Bloom Filters

Figure 10: World Cloud of Cooking Posts before applying Bloom Filters

Figure 11: World Cloud of Sports Posts after applying Bloom Filters

Figure 12: World Cloud of Sports Posts before applying Bloom Filters

Figure 13: World Cloud of Other Posts after applying Bloom Filters

Figure 14: World Cloud of Other Posts before applying Bloom Filters

observed that there are similar changes in both, with the Bloom Filter having slightly more changes.

10 Code

The code associated with this report and assignment can be found on the GitHub Repository - https://github.com/sachinprasanna7/Data-Analysis-of-Tumlr-Posts

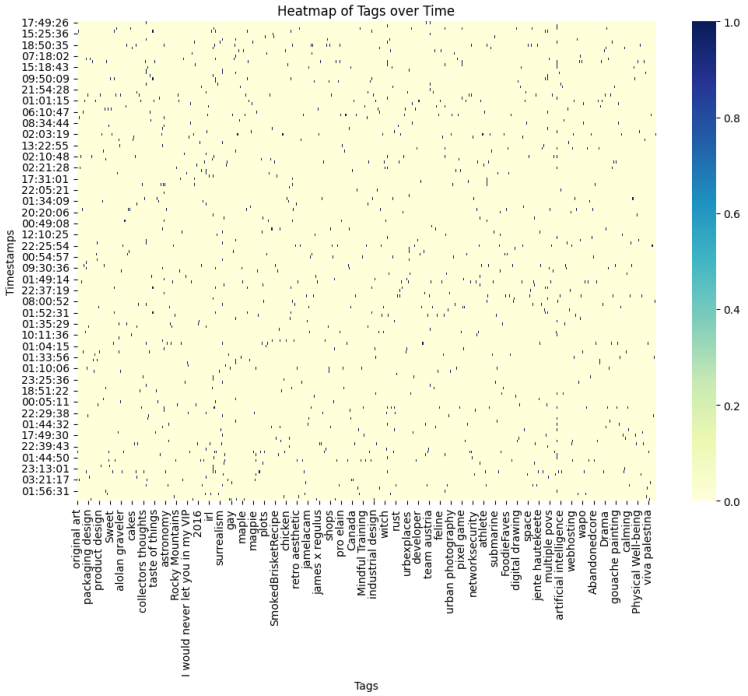

Figure 15: Heatmap of Tags over time with application of Bloom Filter

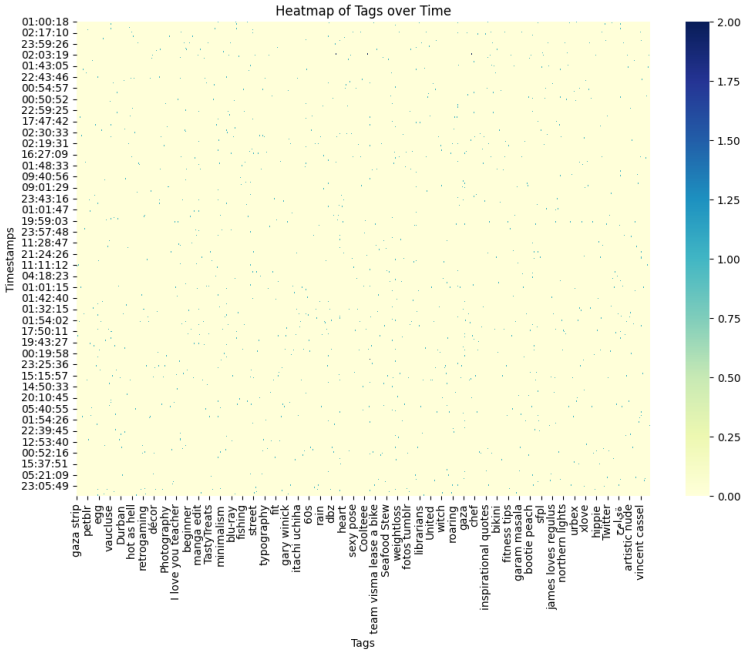

Figure 16: Heatmap of Tags over time without application of Bloom Filter