University of Ontario Institute of Technology
Department of Electrical, Computer, and Software Engineering
Faculty of Engineering and Applied Science

SOFE 3700
Data Management Systems
Final Project

Make Twitter Great Again

Group 17

Member(s)		
First Name	**Last Name**	**Student ID**
Georgy	Zakharov	100588814
Justin	Kaipada	100590167
Savan	Patel	100583384
Karn	Bhavsar	100557957

Abstract

In this report, we discuss our work on creating a database that stores the tweets of the elected president of the United States of America, Donald Trump. These tweets are often viewed as controversial and produce sometimes spectrally different reactions from the audience, depending on the media platform conveying these tweets. Our project aims to provide users with the tool to retrieve original tweets and the information and metadata on these tweets from Twitter and display this data in a flexible way that is inclusive of all the different views and discussions on the tweets across the internet.

This project aims to create a tool that collects and stores data which can be viewed and consumed by the user in an effort to support a particular frame of mind, opinion or discussion, this is achieved through a simple yet functional tool that delivers a number of views of the data.

Introduction

Media and social networks have become the news portals for millennials and the younger generations. These platforms are the main source of news for many people, and they have promoted themselves as reliable sources of information. We get our news from these media platforms and social networks, but what if each organization has its own agenda that goes beyond delivering raw facts? Our news are not raw facts - most of the time and across all major social platforms, the news are subjective opinions of events or facts. The news are biased and are delivered to get a reaction from the audience, rather than to keep the audience informed and equipped with all the facts to make their own unswayed decision.

Since it is hard to change the way social media works at its core, we the people, the consumers of these subjective news need to have a way to analyze through the bias that comes with the news. Our proposed solution is to aggregate all available data on a subject from all different media platforms (starting with Twitter) and make an informed opinion from an overall picture. So far, there have only been archives and web pages storing Trump's tweets, we aim to provide the user with a variety of views that they can use to gain a different point of view on already available data that can provide a full picture from the many different pieces.

Our proposed solution provides a robust, reliable and simple interface to the data and a locally hosted database which can be updated and secured by the user. This is achieved through the use of several modern frameworks like flask, alchemy, pandas as well as modern languages like HTML5 and CSS3.

Main body of work

For this project our target is to collect data about Trump's tweets and store metadata on each tweet along with the tweet content. This information can then be accessed through the front-end API. To achieve this, we will be using HTML5 + CSS3 for the front-end and Python + JavaScript for the back-end. We are accessing Twitter's API in order to scrape the raw data into a .csv file which will be read by a population script to fill up the database; PostgreSQL + pgadmin3 are used to build and maintain the database.

During implementation our difficulties with the front-end were mainly with hosting and formatting, because none of us have had experience with graphical design we needed to opt for a simple and functional design rather than anything professional-like and graphically impressive. With regards to hosting, we chose to forego this route due to time constraints. During back-end development, we were challenged with figuring out the different frameworks needed to put together the functionality for the data scraping and populating of the database. We initially looked to use JSON, but the querying of the Twitter API and JSON use proved more difficult than simply storing the raw data in a .csv, so we chose the latter method.

If we had the opportunity to work on a similar project again, we would do a few things differently. Namely starting earlier in the semester with familiarizing ourselves with the different frameworks, this would allow us to make significant progress quicker when writing, editing and modifying the code. Additionally, we would look into finding royalty-free design templates to be used for the front-end display in order to achieve a more clean and professional look. Lastly, we would look into hosting the website to allow users access to the database through the web instead of local setups.

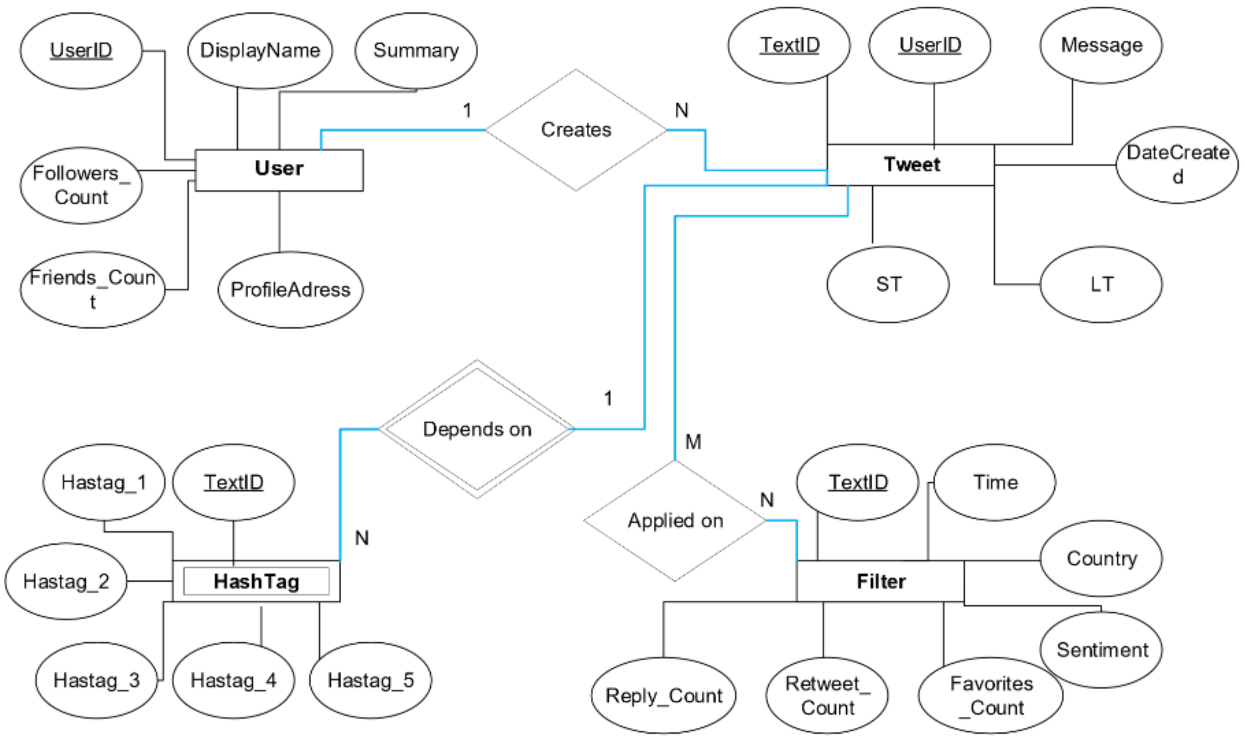
Conclusions with contribution made

Overall, we are satisfied with the quick and effective access of the data from the Twitter API and the wide selection of data that we could choose from. This wide selection allows us to provide different views of the data which can be useful to see trends and patterns and provides tools for objective analysis of tweet content.

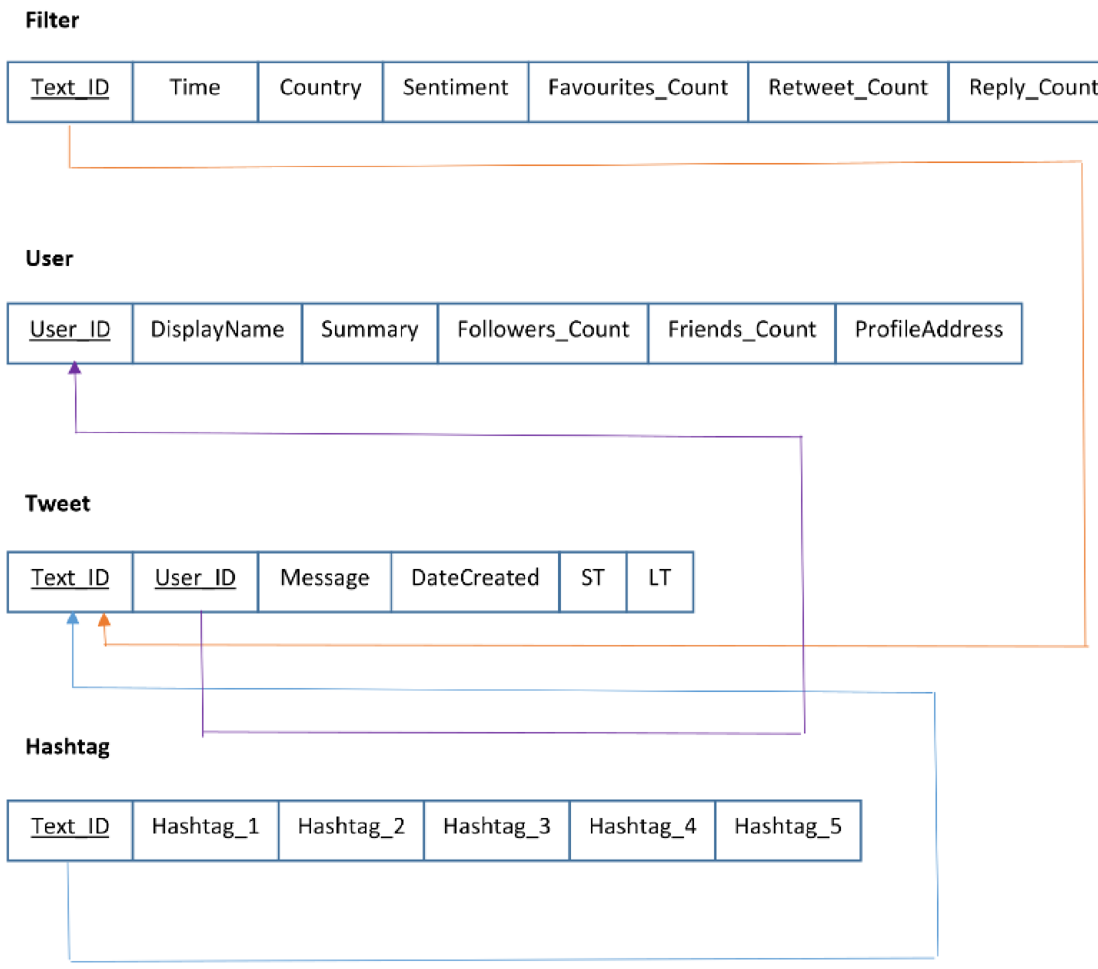
Some inconsistencies may be expected in the future if Twitter significantly changes its API functionalities.

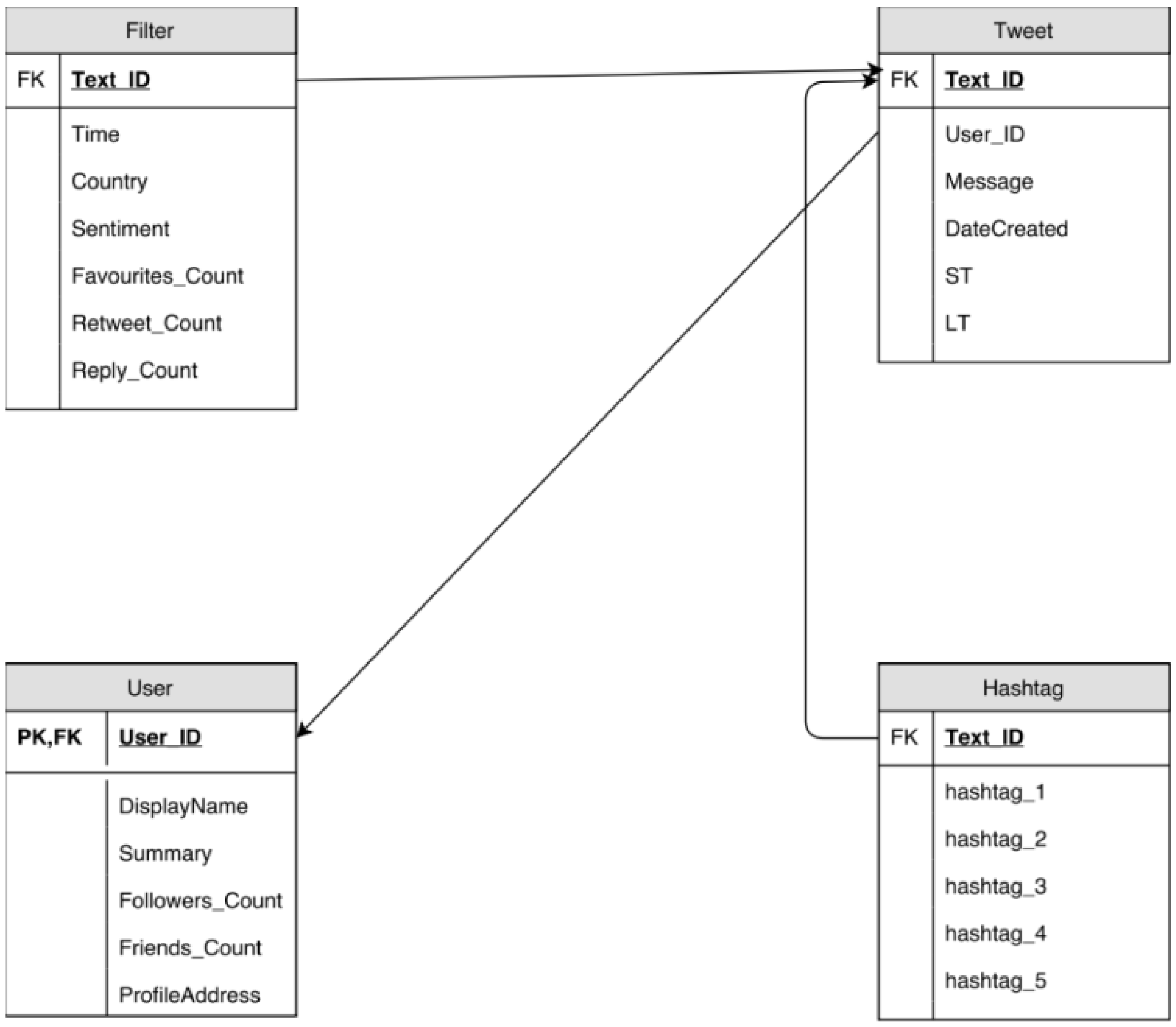
Schematics

ER Schema Diagram



Design diagrams




Filter
FK
Text_ID
Time
Country
Sentiment
Favourites_Count
Retweet_Count
Reply_Count
Tweet
FK
Text_ID
User_ID
Message
DateCreated
ST
LT
User
PK,FK
User_ID
DisplayName
Summary
Followers_Count
Friends_Count
ProfileAddress
Hashtag
FK
Text_ID
hashtag_1
hashtag_2
hashtag_3
hashtag_4
hashtag_5

View 1: Computes a join of at least three tables

```
SELECT *

FROM Filter as F INNER JOIN Tweet as T

ON F.Text_ID = T.Text_ID

INNER JOIN Users as U

ON T.User_ID = U.User_ID
```

View 2: Uses nested queries with the ANY or ALL operator and uses a GROUP BY clause

```
SELECT DisplayName

FROM Users

WHERE User_ID = ANY (SELECT User_ID FROM Tweet WHERE User_ID > 100000)

GROUP BY DisplayName
```

View 3: A correlated nested query

```
SELECT AVG (Favourites_Count)

FROM Filter, Tweet

WHERE Filter.Text_ID = Tweet.Text_ID
```

View 4: Uses a FULL JOIN

```
SELECT st

FROM Tweet

FULL OUTER JOIN Users

ON Tweet.Text_ID = Users.User_ID
```

View 5: Uses nested queries with any of the set operations UNION, EXCEPT, or INTERSECT

```
SELECT User_ID FROM Tweet

UNION

SELECT User_ID FROM Users

ORDER BY User_ID
```

View 6: Computes average retweets count

```
SELECT AVG (Retweet_Count)

FROM Filter, Tweet

WHERE Filter.Text_ID = Tweet.Text_ID
```

View 7: computes highest reply count

```
SELECT COUNT (DISTINCT Reply_Count)

FROM Filter
```

View 8: Finds highest amount of tweets in a single day

```
SELECT COUNT (DISTINCT Text_ID)

FROM Tweet

GROUP BY DateCreated
```

View 9: Latest tweet

```
SELECT *

FROM Tweet

ORDER BY DateCreated DESC

LIMIT 1
```

View 10: Count total tweets

```
SELECT COUNT (Text_ID)

From Tweet
```

View 11: Message containing China

```
SELECT message

FROM Tweet

WHERE message LIKE ‘%China%’
```

View 12: Total Tweets

```
SELECT COUNT (Text_ID)

From Tweet
```

View 13: Get Tweets from User ID

```
SELECT *

FROM Tweet

WHERE “Text_ID” = %s’ , [Tweet_ID]
```

Thoughts about future work

Possible future improvements include providing additional users and their tweets to the database (beyond president Trump), more information and metadata per extracted tweet, and expanding on the types of information offered by implementing a photo/video storage solution. Another possible avenue of improvement may be to create customizable views where the user can select parts of the available data to deliver strong evidence to support their frame of mind.

References

For python templates of scripts

https://github.com/sealneaward

For times when we need to understand black magic

https://stackoverflow.com/

Github Link

https://github.com/justinjk007/Data-Management-Systems-Project-2017