pytwanalysis: Twitter Data Management And Analysis at Scale

Lia Nogueira, Jelena Tešić
Department of Computer Science
Texas State University, San Marcos, TX
l_n63@txstate.edu, jtesic@txstate.edu

***Abstract*—Trends and communities in social media networks shape news cycles, politics, public governing, and economy these days. There is a wealth of information in the way users interact in the large social media networks, and state-of-the-art of mining network data from e.g. Twitter platform is limited by the narrow field of research or computing power. In this paper, we describe the new end-to-end Twitter network data management pipeline. We propose a scalable way to gather, store, and model rich relationships from Twitter networks. We also propose to analyze Twitter data using a combination of graph-clustering and topic modeling techniques at scale using multiple data science methods for graph construction and tweet data processing. We evaluate the proposed system on over 9 million tweets over five different Twitter datasets. We invite the community to add more features, as this end to end pipeline is released as an open source *gitHub repository pytwanalysis* [1], and as a *python pip package pytwanalysis* [2].**

***Index Terms*—Graph Construction, Social Network Management, Graph Analysis, Community Discovery,**

I. Introduction

Social media platform enables users to discuss news topics and trends with friend groups or communities of people. It also allows people to connect on common interests, share information, and influence each other all over the globe. The large number of users, and even larger number of content units (posts, tweets) commentated on and propagated create interesting database for researchers to mine and bring important insights about human behavior, marketing, linguistics, industry trends, brand monitoring, politics, etc. The Twitter platform provides a fast condensed exchange of data, information, and opinions. We have witnessed how much Twitter's discussions and topic trends shape political and commercial campaigns, public policy issues, and marketing strategies for businesses and governments. It is not a straightforward path: information filtering on all social media platforms presents users with the content items most likely to generate further engagement, and users do not know what gets edited out. The rise of social media filter bubble effect makes it harder to separate useful information in social media data from irrelevant data. Current tools used for Twitter analysis focus on the specific project, tend to analyze the data inside a well defined bubble, and fail to generalize the end-to-end process.

In this paper, we propose to scale the efforts on network analysis beyond topic, time, or user group focus. We introduce a scalable way to gather, discover, analyze, and summarize joint sentiment of the Twitter communities. The proposed approach allows researchers to guide the inquiry and data gathering from Twitter, and graph network construction. The contribution of the paper is the data science processing pipeline for Twitter data and a suite of graph-clustering and topic modeling techniques improved for large datasets, and robust content analysis of noisy communities on Twitter released as (i) an open source code [1] and (ii) a software package [2] for general use. We introduce a way to gather, store, and model rich relationships from Twitter networks at scale. We innovate the state-of-art analysis of Twitter data and combine graph-clustering and topic modeling techniques at scale.

Section II summarizes state-of-the-art efforts in social network analysis, graph clustering, topic modeling, and Twitter data processing at scale. There are four major parts of the pipeline, as outlined in Figure 1. *Data Management* presents data science aspects considered for the *pytwanalysis* design: managing semi-structured tweet data at scale, and specific improvements to data cleaning, ingestion, interpretation, and storage to improve efficiency, as described in Section III and illustrated in the data management block in Fig. 1. *Network Analysis* focuses on graph network construction, graph analysis, and visualization. First, we propose methods for multi-modal network construction where an edge in the graph can be constructed from a variety of connection modes (e.g. retweets, replies, quotes, mentions, and hashtags). Next, we propose method for community discovery at scale that combines aspects of several graph-based community detection approaches. Finally, we introduce a suite of visualization and graph reduction techniques to help interpret the meaning of the resulting community labels, as described in Section IV and illustrated in the network analysis block in Fig. 1. *Topic Analysis* addresses the semi-structured nature of tweets e.g. short, slang-, acronym-, and symbol- rich modes of expression, bursting with misspellings, icons, and symbols in Section V, illustrated by the topic analysis block in Fig. 1. Section VI focuses on the reproducibility steps for anyone using the released software package *pytwanalysis* [2] to execute the pipeline steps: tweet extraction, cleaning, storing of tweets in MongoDB, graph-clustering, topic discovery, and visualization. Data Visualization block in Fig. 1 summarizes the pipeline visualization features, and examples of the results are described in Section VII for five different Twitter databases [3] of over 9 million tweets combined.

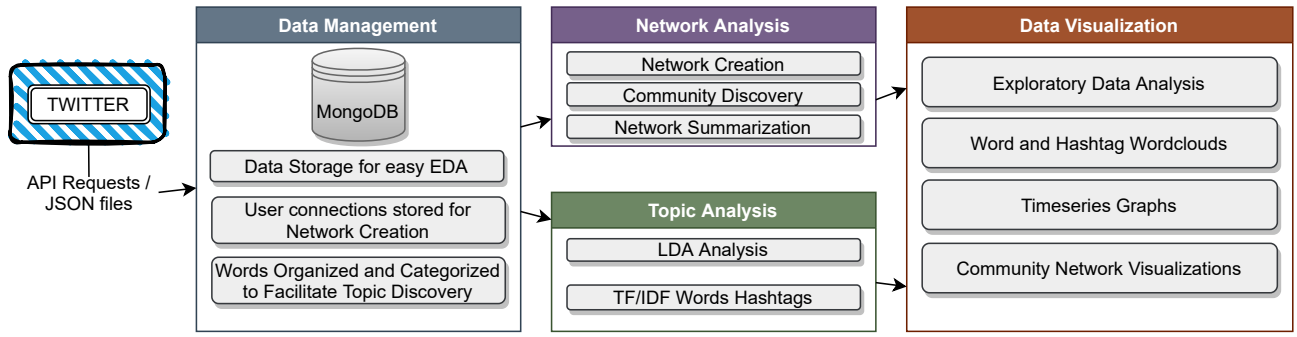

Fig. 1. *pytwanalysis* diagram illustrates all the steps and methods available for tweet extraction, cleaning, storing of tweets in MongoDB, graph-clustering, topic discovery, and visualization [2].

II. Related work

Twitter social media platform influence compelled the data science research to focus on social media analysis. The bulk of the research is focused on the message and the user, i.e. content of the tweets and communication trends among users on the platform [4]. Research work to date has used trends in Twitter for detecting communities and opinions about climate change [5], to understand the influence of fake news in Twitter during the 2016 US presidential election [6], analyze the COVID-19 and the 5G Conspiracy Theory [7], and the COVID-19 Twitter narrative in the U.S. government [8]. Twitter communication trends, generated content, and interactions influenced a variety of aspects of social life, and researchers focus on *community analysis and topic discovery* to model public opinion in social media [9], geography [10], health [11], linguistics [12], economics [13], government policy [14], and political consumerism [15]. Visualization tools to date offer useful visualizing options for existing graphs [16], but can demonstrate limited Twitter pipelines. NodeXL [17] for example, is limited by OS and data size. Related work to date has addressed singular aspects of the end-to-end Twitter data analysis pipeline. Here, we create a scalable unified framework for end-to-end gathering, processing, storing, multi-view analysis, community discovery, topic analysis, and visualization tools that can support different directions of Twitter research data *at scale*. The end user can easily use existing pip package [2] to analyze the rich subspace of twitter-verse of interest, and it can enrich the existing pipeline's open source code [2] with new features for one or many pipeline tasks.

III. Data Management

A tweet document is comprised of multiple objects. Objects retrieved at different dates and with different APIs can have a different set of fields; e.g. deprecated fields still exist in old retrieved files. The tool to manage such data must be able to adapt to the dynamic nature of the Twitter data. We propose to implement a NoSQL solution using *MongoDB*. MongoDB is a schema-less document-oriented database that has shown to scale well with data size in terms of inserts, updates, and simple queries [18]. MongoDB does not need to have a structure limit. That allows for flexibility of the database implementation for Twitter data, as the fields are dynamic in Twitter documents. Researchers have used MongoDB for Twitter data analysis [19], [20], and for the limited scope of storing unstructured textual data [21]. MongoDB does not offer simple functionality for joining collections, and MongoDB databases do not perform well on aggregation queries [18], [22]. In this paper, we propose to use MongoDB and take advantage of the flexibility with unstructured data, and to create multiple data aggregation collections to address aggregated data retrieval and joins. The proposed approach scales and facilitates analysis while improving data retrieval performance.

The Data Management block is in charge of extracting data from Twitter. It supports two data acquisition modes: (i) collect data using Twitter's API and (ii) collect data from offline Twitter JSON objects. For (i), it supports calls to three different endpoints of the Twitter's Search API: the 7-Day search product from the Standard API, the 30-day search product from the Premium API, and the Full-archive search product from the Premium API. The Data Management block also controls all the pre-processing of the data and storage in the database. It saves the raw data extracted from Twitter as well as derived versions of the data that are used to help with the network and topic analysis.

A. Data Collections

MongoDB has the concept of collections, which are a grouping of MongoDB documents. Collections are similar to tables in RDBMS systems, but in MongoDB a schema is not enforced. Even though the schema doesn't have to be rigid, a collection usually stores data with a similar purpose. We propose the use of several collections for data analysis and visualization, and to speed up aggregation queries. We create a handful of collections to store cleaned and transformed data in the data pipeline process. Few of the collections share similar fields by design, as it enables post-processing queries to run easier and faster.

The collections are divided into four types: **Core Collections**: Collections that store the core information needed for analysis. Collections in this group include the *tweet*, *focusedTweet*, *tweetWords*, *tweetConnections*, *tweetHTConnections*, and *users* collections; **Aggregate collections**: Collections containing pre-aggregate data to help with EDA. Collections in this group include the *agg_tweetCountByFile*, *agg_tweetCountByLanguage*, and *agg_tweetCountByMonth*; **Administrative collections**: Collections that save metadata to help administering the data being loaded into the database. Collections in this group include the *adm_loadStatus*,

adm_loadedFiles, and *searches* collections; and the **Temp collections**: Auxiliary collections used to improve the complex data retrieval queries. Collections in this group include the *tmpEdges*, *tmpEdgesTweetIds*, *tmpEdgesHTFreq*, and *tmpEdgesWordFreq*. The collection *tweet* stores the raw data from the Twitter JSON files, with no modification, and the collection *focusedTweet* stores the focused data extracted from the original documents. It is useful to have a separate collection for the focus data as it (i) decreases the amount of data stored in one collection and increases the query's performance, and (ii) enables core columns with a standard name processing. The field that contains the original tweet message can have different names, and standardizing that name in the focused collection streamlines the pipeline processing. Details of the data dictionary and description of every field and collection in the proposed database are available in the package documentation [2].

B. Pipeline Performance Features

In order to deal with the performance challenges while inserting and retrieving data from the collections, we propose 4 different improvements in the pipeline: **1. Recovery Process:** The pipeline includes a recovery logic to make sure that the processes doesn't have to be re-run from the beginning in case of a failure. While inserting into the *tweet* collection, a sequence number gets attached to each tweet. Every time any processes need to run, that sequence number is used to control what has already been processed or not. If something fails, the logic will be able to identify the last sequence number processed and continue from there. That logic is available for all the core collections and is driven by the values stored on *adm_loadStatus*. **2. InsertOne vs InsertMany:** MongoDB gives users the flexibility to insert records one by one using *insertOne* method, or insert multiple records at a time using *insertMany*. *insertMany* method scales better for large datasets, but we need to store all data in the memory first. We propose the solution to insert data in large groups: retrieve, process and save record in the memory, and when N records are saved call $insertMany$ and write them to MongoDB. To avoid both problems and take into consideration the differences in hardware specification, in *pytwanalysis* package N is a configuration parameter, which is the maximum number of records to be inserted at any given time. This allows the system to be used on anything from laptops (smaller N) to workstations (larger N). **3. Indexing:** Aggregation, filtering and retrieval processes in MongoDB databases can cause a bottleneck depending on the amount of data and filters used. The *pytwanalysis* package improves the processes by automatically indexing strategic fields at database creation, e.g. seq_no (a numeric sequence value used in the recovery process), tweet_id_str (an unique id from Twitter application that identifies each tweet), user_id_str (an unique id from Twitter application that identifies each user). The times to execute queries and to create the derived collections (results of aggregation or filtering) has significantly improved with index support. **4. Temporary Collections:** Not all aggregations and queries can be sped up with indexing. For a query to retrieve hashtags that were used by users from specific edges in a graph, it will take way too much time to execute the query for a large graph network. We propose the following: (i) create a temporary collection with all the edges for that specific graph; (2) create temporary collection that saves all the tweet Ids for those edges. Now, the resulting data is easily aggregated to count the hashtags used for those tweets. After implementing these methods, the average execution time in *pytwanalysis* decreased by 60%.

IV. Graph Networks Creation, Analysis, and Summary

Network Analysis block in Figure 1 consists of 3 sub-blocks: Network Creation (Sec. IV-A), Community Discovery (Sec IV-B), and Network Summary (IV-C).

A. Network Creation

The *pytwanalysis* package creates multiple undirected networks for the same set of Twitter records as shown in Figure 2. The edges of the networks are saved in the collections *tweetConnections*, and *tweetHTConnections* (section III-A). We adopt Twitter users and hashtags as vertices, and create different networks based on what is interpreted as an edge's information.

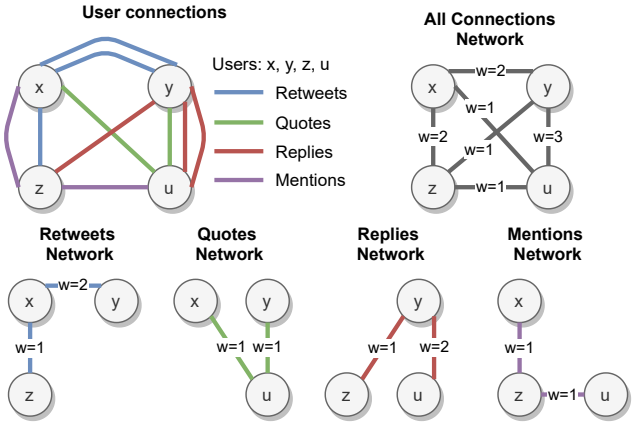

Fig. 2. Networks created from the different user connections.

The **Retweets Only** network method inserts a weighted edge between users based on their retweets. For two Twitter users, *x* and *y*, edge *(x, y)* gets created in *G* when either *x* retweets a tweet from *y*, or *y* retweets a tweet from *x*. The *weight(x, y)* will be the total number of retweets from each direction. Vertices with a high degree will represent users that were retweeted frequently. The **Quotes Only**, **Replies Only**, and **Mentions Only** network methods are created similarly to the *Retweets Only* network, but with each weighted edge being the connection between two users by quotes only, reply only, or mentions only, respectively. In the the mentions only network, even if a user *y* doesn't tweet, retweet, or reply to anybody, they can still show up as a high degree vertex in the mentions network in case thousands of other people mentioned them in their tweets. The **All User Connections** network method combines all user connections together: retweets, quotes, replies, and mentions. Edge weight is the total number

of retweets, quotes, replies, and mentions interactions. The **Hashtag Connections** network creation method identifies hashtags that are frequently used together. Let *#x*, *#y*, and *#z* be the hashtags used in a single tweet. The hashtag connections network method will create the pairs {*#x, #y*}, {*#x, #z*}, and {*#y, #z*} for this single tweet. Edge weight indicates how many times two hashtags were used together.

B. Community Discovery

We use three different methods of clustering for analysing the data and evaluate the performance of the methods using 4 primary and 3 secondary measures.

Community Discovery Methods: • **Louvain Community Detection** method [23] is a heuristic method based on modularity optimization that has been shown to outperform other known community detection methods in terms of computation time without losing quality. The *Louvain* method is a greedy optimization method with a runtime that increases close to linearly with the number of vertices in the network, and it has exhibited good summarization power compared to other methods [24]. The outcome of the *Louvain* Community Detection is a set of disjointed communities with densely connected vertices. • **Spectral Clustering** approach performs the *kmeans* clustering method on k eigenvectors corresponding to the smallest eigenvalues (not 0) of the Laplacian of the adjacency matrix of the graph. Laplacians can be defined in multiple ways, and based on that, there exist many different implementations. *Spectral Clustering* depends on the k, the number of clusters desired, and we select *k* that maximizes the eigengap [25]. The output is a set of disjointed communities with densely connected vertices. The method has a high computation time and can only be used in fully connected networks, as it uses a similarity matrix to identify how similar the vertices are. We use the method *SpectralClustering* from the *sklearn* [26] package, and in the output each vertex gets assigned to a cluster label. To transform the graphs into adjacent matrices that can be sent as parameters to the *SpectralClustering* method, the *to_scipy_sparse_matrix* method from the *networkX* package [27] was used. • **Top k Degree Vertices' Neighborhoods (TN-Neighborhoods)** *Super users* shape Twitter communities and guide topic discovery and development, as 80% of all tweets come from just 10% of all U.S. Twitter users [28]. These *super users* tweet 138 times per month, and the median Twitter user tweets twice per month [28]. Super users are found to be active in multiple communities, as community connections vary. Graph-theoretic relaxation of the concept of cluster graphs introduced overlaps between the clusters [29]. We propose a novel cluster overlap approach for *building communities around super users* with relaxation criteria that a user can belong to multiple communities. The influence of vertex v is defined by its degree in the graph, and we consider the vertices with the degree higher than k as new community centers of the graph. For a network graph *G*, and a vertex *v*, $v \in V \wedge |v| > k$, create a neighborhood sub-graph $N(v)$. The sub-graph will contain all vertices that are connected to v and all the edges that connect the vertices in the sub-graph together. We grow these communities around the highest degree vertices in the dataset. Since users can be connected to multiple high degree vertices, the result of this method is a set of overlapping communities, and the communities capture discussions initiated or related to a set of high influence vertices *v*. The visualization of high level steps of the approach is illustrated in Figure 3.

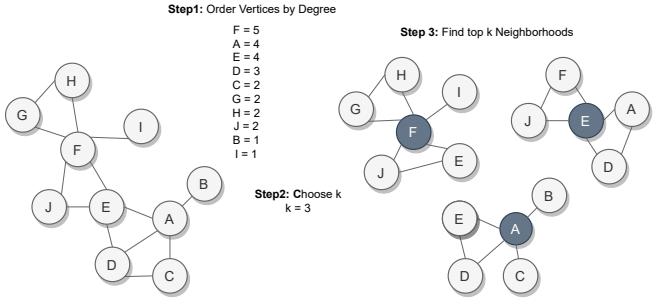

Fig. 3. Visualization of the high level steps of the *TN-Neighborhoods* algorithm.

Measures of Performance: We use 4 measures of performance for the community detection: *Separability*, *Density*, *Clustering Coefficient* [30], and *Power Nodes Score* , and 3 graph characteristics measures: the average degree of vertices, graph clique size, and number of cliques. • **Separability** measures the ratio between the edges of the community (internal vertices) and the edges of the vertices of the community that are pointing to the outside, under the assumption that solid communities would be well-separated from the rest of the network. Let *G[V]* be the induced sub-graph of graph *G* for vertex subset *V*, where *V* was created from one of the clustering methods. And let E_m be the number of edges in *G[V]*, and E_c the number of edges on the boundary of *G[V]*. Then we calculate separability as $sep = \frac{E_m}{E_c}$. **Density** measures how well connected the vertices are. Let *G[V]* be the induced sub-graph of graph *G* for vertex subset *V*, where *V* was created from one of the clustering methods. And let E_m be the number of edges in *G[V]*, and V_n the number of vertices in *V*. Then we calculate density as $den = \frac{2*E_m}{V_n*(V_n-1)}$. **Power Nodes Score** identifies graphs that are highly connected through a set of highly connected vertices. A high score will indicate that the top degree vertices of the graph are highly connected to all other vertices in the graph. Let *G[V]* be the induced sub-graph of graph *G* for vertex subset *V*, where *V* was created using the *TN-Neighborhoods* approach for the highest-degree k vertices. If V_n is the number of vertices in *V*, and *n* the number of vertices in the original graph *G*, the power vertices score is calculated as $pns = \frac{V_n}{n}$. Note that *k* is the number of clusters for all clustering methods, and we use the same k in all methods for comparison. **Average Clustering Coefficient** is the average of the local clustering measures for each vertex. The local clustering of each vertex in *G* is the ratio between the triangles that actually exist and all possible triangles in its neighborhood [31]. It was proposed as a goodness metric on the premise that pairs of vertices with the same neighbors are likely to be connected to each other. **Average Degree of Vertices** quantifies the average degree of all vertices in G, and

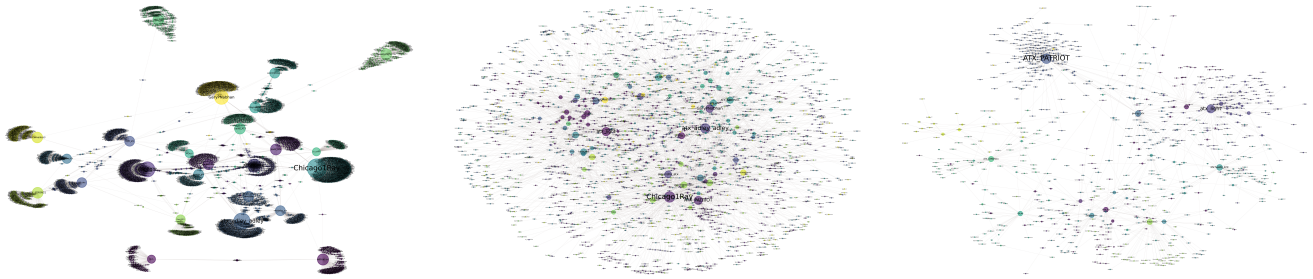

Fig. 4. Graph reduction samples from the *Austin* dataset. *Vertex Degree* graph reduction outcome: filtered all edges connecting two vertices with degree less than 115 (left); *Community Percentage* and the *Vertex Degree* graph reduction outcome: every community found in the graph reduced by 70%, and then every edge with degree less than 25 removed (center). *Edge Weight* and the *Community Percentage* aggregation: every edge with weight 1 removed, followed by the community reduction by 2% (right).

measures graph connectivity. Let V be the vertex-set of G, and n the number of vertices in V. Then calculate the average degree of vertices as $av_n = \frac{\sum_{i=1}^{n} V_i = deg(V_i)}{n}$. **Graph Clique Size** measures the number of vertices in the largest clique of G. A clique C in an undirected graph G, is the induced sub-graph of G where every two distinct vertices in C are adjacent. In social networks, cliques could represent groups of people where everyone in the group has a connection with every other person in that same group. **Number of Cliques** is the number of maximal cliques in G.

C. Network Summary

Large networks contain millions of vertices that connect to each other in some way [9]. Community discovery allows us to analyze one community at a time. When a community contains tens of thousands of vertices, visualizing the community as an entity does not provide a lot of information. Grouping of vertices enables us to conduct meaningful community analysis. The specific technique employed depends on the research question we are trying to answer, what will have a low cost if discarded, and what the aggregation is meant to accomplish, as they can focus on graph aggregation to maintain the same structure, identify patterns, maintain the most salient information of the original graph, or focus on the most influential vertices. Various graph aggregation approaches have succeeded in reducing the graphs without major structural changes, while reducing the high processing cost [24]. In this paper, we propose and implement context-aware graph reduction methods that allow for fast and meaningful visualization: • **Community Percentage:** A percentage of vertices of every community found in a given graph is removed, starting from the lower degree vertices. This approach reduces the number of vertices to plot, without changing the overall structure of the data. The *Louvain* method was the chosen algorithm to calculate the communities, as it has been shown to have good summarization power and excellent execution time [24]; • **Edge Weight:** Given a number x, we remove all edges (u, v) with a weight less than or equal to x, $weight(u, v) <= x$. Removing "weak links," edges with a small weight allow a visual analysis of vertices with stronger connections; and • **Vertex Degree:** Given a number x, we remove all vertices v with degree less than or equal to x. Removing edges related to low degree vertices will allow the visualization to reveal vertices that interact with more vertices. Figure 4 shows three graphs created from the exact same vertices and edges, but using different reduction techniques.

V. Topic Analysis

•**Pre-Processing Methods** Tweet text was cleaned before it was used for topic modeling as follows: (i) Remove links by cleaning everything that starts with *http*, (ii) remove hashtags by cleaning everything that starts with #, (iii) remove mentions by cleaning everything that starts with @,(iv) remove retweet symbols at the beginning of retweet messages, (v) remove any symbols, punctuation, return characters, extra spaces, special characters, and numbers, (vi) remove stop words, and (vii) lemmatize words. •**Topic Model** We have used the Latent Dirichlet Allocation (LDA) [32] module from the *gensim* package [33] to train the topic model: (i) create a set of documents where each document corresponds to one tweet, (ii) pre-process the content of each document as explained in *Pre-Processing Methods*, (iii) create a term dictionary of the corpus, where every unique term is assigned an index, (iv) convert the list of documents into a *Document Term Matrix* using the dictionary prepared in the previous step, (v) choose the number of topics, (vi) and train the model using the *gensim* package [33]. Given a set of documents, the LDA model returns a set of topics, each with a set of words with a probability score that shows how likely each word can describe the topic. •**Evaluation** *pytwanalysis* package implements the coherence metric *c_v* as an automated measure of evaluation in a large number of experiments as it was shown to outperform other existing coherence metrics [34]. •**Aggregation Methods** People use the same hashtag for different discussions threads. The LDA model does an adequate job classifying topics for well defined documents, where the colloquial nature of Twitter data poses a challenge. To mitigate this issue, *pytwanalysis* pipeline pre-aggregate the tweets as the first level of topic discovery process using three different methods: (i) graph-based hashtag communities, (ii) graph-based user connection communities, (iii) and filters by time period, user's screen_names, or hashtags. The graph-based *hashtag communities aggregation* uses one of 3 methods from Sec. IV-B to create a network of hashtag connections and separate them using the clustering techniques. Even though the hashtags themselves can be difficult to interpret, they can be

great indicators of topics. The graph-based *user connections aggregation* uses one of 3 clustering methods from Sec. IV-B to create a network of user connections and separate them using the clustering techniques. Users that are very connected to each other tend to share similar topics. The *filters* aggregation separates the tweets into groups that are bound based on the filter used. Time period filter, for example, separates the dataset into tweets that were created within a date range. That way it will be possible to see the topics being discussed for that specific period. We have designed the pipeline so that the three aggregation techniques can be used as stand-alone or in combination. Note that the *pytwanalysis* pipeline parametrised the choice of multiple different network creation algorithms, community discovery algorithms, and topic modeling aggregation methods. An example of this modular pipeline selection is to aggregate the tweets based on the *Louvain* communities found in the replies only networks for a certain time period.

VI. Pipeline Features

The *pytwanalysis* package provides a streamlined way for users to access the functionalities of all pipeline blocks. Twitter data analysis pipeline consists of several blocks, as illustrated in Figure 1. In the *Data Management* block, tweets are downloaded from the source and saved as JSON objects. The Twitter JSON objects are cleaned, organized, and loaded into a MongoDB database. Next, in the *Network Analysis* block, network creation, community discovery, and graph summarization and visualization are available. The *Topic Analysis* block supports topic discovery using LDA, and word and hashtag frequency analysis. The *Data Visualization* block combines the analysis of all previous block and outputs the analysis of the data. It outputs exploratory data analysis; files with word and hashtags frequency analysis; wordclouds, barcharts, and timeseries visualizations explaining the data; networks analysis files including community separation, graph visualization, and vertices and edges details; and full analysis automation that allows the user to run a full analysis with all components without having to run any of the analysis individually. The pipeline is subject agnostic: data acquisition and processing are determined by the end user based on the research question they try to answer. The *pytwanalysis* package is available for installation at pip [2], the code is available in gitHub [1], along with the detailed documentation. The *pytwanalysis* package contains 4 classes: (i) **TwitterDB**: This class will take care of all the MongoDB activities. It will load tweets from JSON files, create new collections and indexes, clean and transform data, query data, and export data; (ii) **TwitterGraphs**: This class will take care of graph related tasks. It will analyze graphs, export graph metrics, create sub-graphs, create clusters, calculate cluster metrics, plot graphs, and reduce graphs; (iii) **TwitterTopics**: This class will take care of topic discovery related tasks. It will train the LDA model, and print graphs with word and hashtag frequency; and (iv) **TwitterAnalysis**: This class will inherit the methods of the other three classes and will be in charge of automating the creation of the analysis files and folder structure. The *pytwanalysis* package enables parametric configuration so multiple methods and approaches can be tested, and it integrates the following packages: networkx [27], scikit-learn [26], gensim [33], NLTK [35], Community Discovery [36]. Details on the package use, data collection, and automated analysis are provided in the documentation [2].

VII. Data Analysis and Results

Dataset	Random	Austin	BJJ-en	BJJ-pt	Covid
Tweet Count	160,066	310,601	242,956	267,101	8,123,104

TABLE I
Datasets used in the experiments

Five datasets were used as case studies for the experiments [3]: **Austin**, a dataset with over 300,000 tweets that are related to the hashtags #austintexas, #atx, #austintx or #atxlife; **BJJ-en**, a dataset in English with over 200,000 tweets that are related to the hashtags #BJJ, #jiujitsu or #jiu-jitsu; **BJJ-pt**, a dataset in Portuguese with over 200,000 tweets that are related to the hashtags #BJJ, #jiujitsu or #jiu-jitsu; **Covid**, a dataset with over 8 million tweets that are related to the hashtags #Coronovavirus, #Covid19 or #Covid-19; and **Random**, a dataset with over 100,000 random tweets that are not related to any particular subject. Next, we run *pytwanalysis* package on the data, and present the results. For each dataset, 6 types of networks are created, as outlined in Sec. IV.

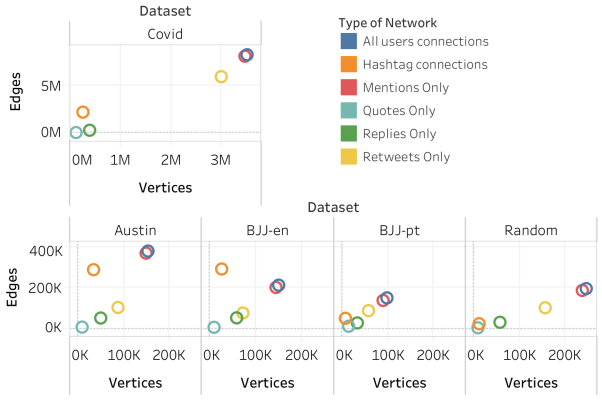

Fig. 5. Subfigures correspond to five datasets: the number of Vertices and Edges for each type of network is illustrated in 2D space as a colored circle.

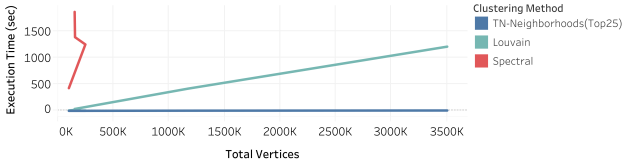

Fig. 6. Execution time in seconds for three clustering methods as a function of the number of vertices in the graph. Spectral clustering fails for large graphs due to known eigenvalue computation issues in the algorithm.

Figure 5 summarizes the number of vertices and edges for each type of network for each dataset. The number of edges for each of the networks seems to follow a pattern, with the smallest network always being the *quotes* network, followed by the *replies* network, then *retweets*, *mentions*, and *all connections*. The richer the network, the higher the number of vertices and edges. Hashtag connection networks are more dense as they have a relatively small number of

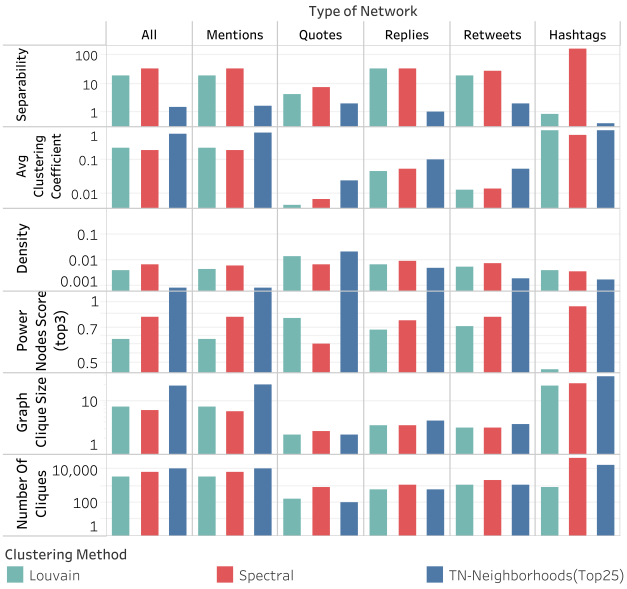

Fig. 7. Metrics average of the communities found using the different clustering methods for *Austin* dataset for the different types of networks.

vertices (frequent hashtags) and a high number of edges. Observe $x = y$ slope on all graphs (number of vertices is close to number of edges). All datasets that were collected using term and topic search are more dense (number of edges is higher) than random dataset collection, as there is no connectivity in the community for random tweets. Next, we apply the three clustering algorithms described in Sec. IV-B on each of the networks. Figure 6 shows a comparison of the execution time of each clustering method. The *Spectral Clustering* method has the highest execution time, followed by the *Louvain* method, and then *TN-Neighborhoods*. Both the *Louvain* and *TN-Neighborhoods* methods seem linear, but the *TN-Neighborhoods* has a much slower slope. The execution time experiments were done on a Windows machine, with the Intel(R) Core(TM) i-9900K CPU @3.60GHz, and 64G of RAM. Figure 7 shows the metrics for the three types of clustering methods for the *Austin* dataset for all types of networks. The hashtag connections network has the highest score for separability and average clustering coefficient, but has lower density. Separability is similar for all other network types. All six types of networks follow similar values for the power nodes score.

Graph Reduction Experiment: We compare graphs before and after the use of the reduction techniques shown in Sec. IV for the Austin dataset tweets in the month of May, using the retweet connections network. The original graph without any reductions has a total of 10,448 vertices and 11,977 edges. We then compare the metrics for the original and reduced graphs: the number of vertices, number of edges, average degree vertex, density, power nodes score, vertices and edges compression percentage, and the top 1% vertices similarity calculations are compared. Fig. 8 shows a comparison of the different reduction techniques and the effect caused on the resulting compressed graph. The number of vertices and edges decreases as the level of reduction for each method increases, as shown in Figure 8. The compression occurs more gradually for the *Community Percentage* method compared to the others. The average degree vertex increases for the *Vertex Degree* method as the graph becomes more compressed. This behaviour is expected, since the vertices that are only connected to lower degree vertices are getting removed from the graph. The opposite seems to occur for the *Edge Weight* method. Since the edges with low weight are getting removed, vertices that had high connectivity because of the pendant vertices will have their degree lowered. The density of the graph increases as the reduction methods are applied, and the top vertices' similarity decreases. Figure 4 shows three graphs created from the exact same vertices and edges, but using different reduction techniques.

Topic Discovery: We used our graph-based aggregation technique shown in section V to find topics within our datasets. Figure 9 shows an example of visualizations of a certain topic found in a community from the Austin dataset.

VIII. Summary and Conclusions

In this paper, we present a scalable way to gather, discover, analyze, and summarize large datasets from the Twitter social media platform. The system foundation is a suite of techniques in social network analysis, and the final product is a scalable end-to-end data science pipeline that adapts to the dynamic and semi-structured nature of Twitter data, release as an open source code [1] and pip python package [2]. The pipeline was designed to be used by a wide range of researchers that are interested in analyzing Twitter data for answering questions on several areas of study. Future work includes the package being expanded with more functionality, such as sentiment analysis, fake news detection, and support for other Twitter APIs.

References

[1] "pytwanalysis source code." https://github.com/lianogueira/pytwanalysis (2021, Oct.).

[2] "pytwanalysis package." https://pypi.org/project/pytwanalysis/ (2021, Oct.).

[3] L. Nogueira, "Social network analysis at scale: Graph-based analysis of twitter trends and communities," Master's thesis, Texas State University, 2020.

[4] A. Bruns, K. Weller, M. Zimmer, and N. J. Proferes, "A topology of twitter research: disciplines, methods, and ethics," *Aslib Journal of Information Management*, 2014.

[5] W. Pearce, K. Holmberg, I. Hellsten, and B. Nerlich, "Climate change on twitter: Topics, communities and conversations about the 2013 ipcc working group 1 report," *PloS one*, vol. 9, no. 4, p. e94785, 2014.

[6] A. Bovet and H. A. Makse, "Influence of fake news in twitter during the 2016 us presidential election," *Nature communications*, vol. 10, no. 1, pp. 1–14, 2019.

[7] W. Ahmed, J. Vidal-Alaball, J. Downing, and F. L. Seguí, "Covid-19 and the 5g conspiracy theory: social network analysis of twitter data," *Journal of Medical Internet Research*, vol. 22, no. 5, p. e19458, 2020.

[8] H. Sha, M. A. Hasan, G. Mohler, and P. J. Brantingham, "Dynamic topic modeling of the covid-19 twitter narrative among us governors and cabinet executives," *arXiv preprint arXiv:2004.11692*, 2020.

[9] J. Leskovec and R. Sosič, "Snap: A general-purpose network analysis and graph-mining library," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 8, no. 1, pp. 1–20, 2016.

[10] L. Zou, N. S. Lam, S. Shams, H. Cai, M. A. Meyer, S. Yang, K. Lee, S.-J. Park, and M. A. Reams, "Social and geographical disparities in twitter use during hurricane harvey," *International Journal of Digital Earth*, vol. 12, no. 11, pp. 1300–1318, 2019.

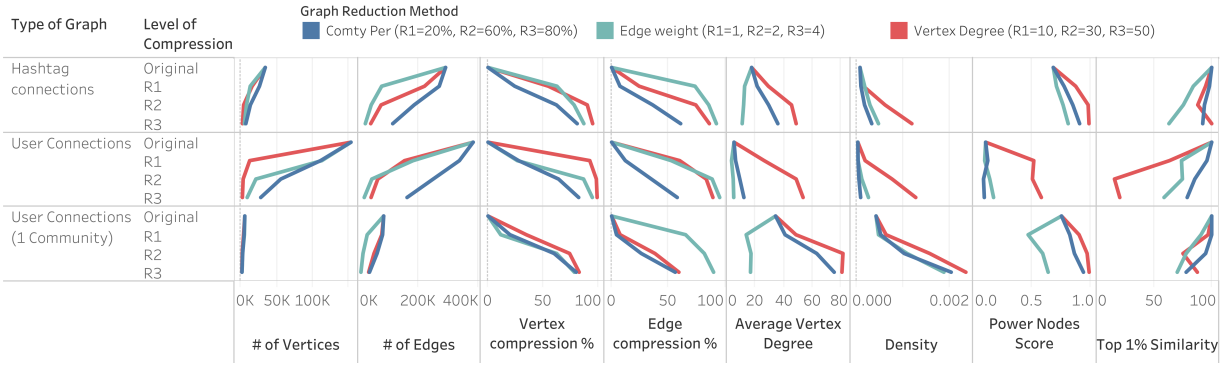

Fig. 8. Comparison of the different reduction techniques for three graphs of different sizes. Data was extracted from the *Austin* dataset.

Fig. 9. Three sample visualizations for a topic discovered using a graph-based aggregation technique: LDA output as a bar chart, word cloud of hashtags used in the community, and graph illustration connecting the hashtags in the community. Data was extracted from a community found in the hashtag network in the *Austin* dataset.

[11] L. S. et. al, "Twitter as a tool for health research: a systematic review," *American journal of public health*, vol. 107, no. 1, pp. e1–e8, 2017.

[12] Y. Huang, D. Guo, A. Kasakoff, and J. Grieve, "Understanding us regional linguistic variation with twitter data analysis," *Computers, environment and urban systems*, vol. 59, pp. 244–255, 2016.

[13] S. Bijarnia, P. Ilavarasan, and A. Kar, "Comparing servqual for transportation services in the sharing economy for emerging markets: Insights from twitter analytics," *Digital and Social Media Marketing. Advances in Theory and Practice of Emerging Markets*.

[14] D. Duran-Rodas, D. Villeneuve, and G. Wulfhorst, "Bike-sharing: the good, the bad, and the future -an analysis of the public discussion on twitter-," *European Journal of Transport and Infrastructure Research*, vol. 20, no. 4, pp. 38–58, 2020.

[15] O. Johnson, A. Hall-Phillips, T.-L. D. Chung, and H. Cho, "Are you connected through consumption? the role of hashtags in political consumption," *Social Media + Society*, vol. 5, no. 4, 2019.

[16] M. Bastian, S. Heymann, M. Jacomy, *et al.*, "Gephi: an open source software for exploring and manipulating networks.," *Icwsm*, vol. 8, no. 2009, pp. 361–362, 2009.

[17] M. Smith, N. Milic-Frayling, B. Shneiderman, E. Mendes Rodrigues, J. Leskovec, and C. Dunne, "Nodexl: a free and open network overview, discovery and exploration add-in for excel 2007/2010," 2010.

[18] Z. Parker, S. Poe, and S. V. Vrbsky, "Comparing nosql mongodb to an sql db," in *Proc. 51st ACM SE Conference*, pp. 1–6, 2013.

[19] C.-S. Atodiresei, A. Tănăselea, and A. Iftene, "Identifying fake news and fake users on twitter," *Procedia Computer Science*, vol. 126, pp. 451–461, 2018.

[20] M. Trupthi, S. Pabboju, and G. Narasimha, "Sentiment analysis on twitter using streaming api," in *2017 IEEE 7th International Advance Computing Conference (IACC)*, pp. 915–919, IEEE, 2017.

[21] P. Kumar, "Analysis of a data processing pipeline for generating knowledge graphs from unstructured data: Data processing pipeline for knowledge graphs," Master's thesis, Delft University of Technology, 8 2020. EIT Digital Double Degree Programme, Epema, D.H.J. (mentor).

[22] A. Celesti, M. Fazio, and M. Villari, "A study on join operations in mongodb preserving collections data models for future internet applications," *Future Internet*, vol. 11, no. 4, p. 83, 2019.

[23] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.

[24] Y. Liu, T. Safavi, N. Shah, and D. Koutra, "Reducing large graphs to small supergraphs: a unified approach," *Social Network Analysis and Mining*, vol. 8, no. 1, p. 17, 2018.

[25] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[26] F. P. et. al., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[27] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conference* (G. Varoquaux, T. Vaught, and J. Millman, eds.), (Pasadena, CA USA), pp. 11 – 15, 2008.

[28] S. Wojcik and A. Hughes, "Sizing up twitter users." https://www.pewresearch.org/internet/wp-content/uploads/sites/9/2019/04/twitter_opinions_4_18_final_clean.pdf, 4 2019.

[29] M. R. Fellows, J. Guo, C. Komusiewicz, R. Niedermeier, and J. Uhlmann, "Graph-based data clustering with overlaps," *Discrete Optimization*, vol. 8, no. 1, pp. 2 – 17, 2011. Parameterized Complexity of Discrete Optimization.

[30] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," *Knowledge and Information Systems*, vol. 42, no. 1, pp. 181–213, 2015.

[31] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Predicting positive and negative links in online social networks," in *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pp. 641–650, ACM, 2010.

[32] D. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[33] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, (Valletta, Malta), pp. 45–50, ELRA, May 2010. http://is.muni.cz/publication/884893/en.

[34] M. Röder, A. Both, and A. Hinneburg, "Exploring the space of topic coherence measures," in *Proceedings of the eighth ACM international conference on Web search and data mining*, pp. 399–408, 2015.

[35] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[36] T. Aynaud, "python-louvain 0.14: Louvain algorithm for community detection." `https://github.com/taynaud/python-louvain`, 2020.