University of Stavanger

Faculty of Science and Technology

MASTER'S THESIS

Study program/Specialization: Computer Science, Reliable and Secure Systems.	Spring semester, 2021 Open access
Writer: Mohammed Zoher Guniem	Mohammed Guniem (Writer's Signature)
Faculty supervisors: Antorweep Chakravorty Nikita Karandikar	
Thesis title: Identifying and Classifying Social Influencers - Engagement Analysis	
Credits (ECTS): 30 Points	
Key words: social-graph, reddit, influence-detection, influence-weight, influence-field-classification, centrality-measures, mongo-databases, neo4j-databases, online-social-media-analysis.	Number of pages: 68 + Source code of the technical solution is attached to this PDF, also available on GitHub via the link: https://github.com/MohammedGuniem/social-media-influence_analyzer + Demonstration website at https://smia.uis.no. Stavanger, 10.08.2021

Table of Contents

Social Media Influence Analyzer

Mohammed Z. Guniem
Department of Computer and Electrical Technology
University of Stavanger
Stavanger, Norway
mghunime@yahoo.no

In a time where online social media has a significant influence on both individuals and groups of people in our modern societies, there is a need to have a digital tool that helps gather and construct data from online social media, then use this data to detect influence between individual users by measuring and classifying the strength and field of this detected influence. This research proposes a method for influence detection, then builds a technical solution on top of it to enable analyzers of online social media to have a visual and scientific understanding of the influence flow in online social media. The results achieved from the project of this research is a technical information system that can crawl data from online social media, then detect and classify influence between online users and visualize this information in intuitive graphs.

Keywords: social-graph, reddit, influence-detection, influence-weight, influence-field-classification, centrality-measures, mongo-databases, neo4j-databases, online-social-media-analysis.

Motivation

Upon the rise of the digital revolution through the last two decades, people around the world are no longer limited to the constraints of place and time to socialize with each other. The newly introduced concept of digital media has transformed the way our society functions. Socializing no longer requires the physical presence of society members, and multiple newly introduced social media platforms are now connecting people from all over the world, and engaging them in local, national, and global events. Platforms of digital media are giving people the opportunity to participate in society and express themself in an open arena where physical boundaries do not stand in the way.

Today and after a very short time of experiencing the advantages of digital media, our society has become almost dependent on such platforms, and most social events and happenings are being recorded and discussed in the wide arena of social media. This effect generates a huge amount of valuable data that has a big potential of revealing the type and strength of social influence between society members and opens for many useful applications in multiple fields.

The most obvious application from social data is understanding how social media is used as a tool to mobilize groups of people in controversial social events such as political elections. The serious allegation of Russian interference in the US presidential election in 2016 is one application of analyzing social influence on social media.

Furthermore, by mapping and visualizing social influence between users of online social media, we can speed up and improve the detection of fake news and other illegal activities on social media, and by removing their damaging effects on multiple social environments, we can create a healthier society that benefits all its members.

Social influence is also highly valuable for commercial use, as many companies are interested in detecting different types of social influence to reveal new marketing trends and allow businesses to develop more specialized marketing strategies and customized products. This increases the competition in the economy and generates more value for companies and their surrounding societies.

These were some applications that can benefit from analyzing the influence between users on social media, and there are still both uncovered and undiscovered areas where understanding social influence is highly crucial for a related application.

I. INTRODUCTION

Data from social media has great potential in revealing how strong the influence is between different users. Just like in real life every action a user commit, and how society members react to this action can serve as an object for analysis. Which helps in drawing a big but rather detailed picture of how users influence each other across many societies and fields.

This research aims to establish a ground foundation for extracting information about user activities from networks of social media. Then use such information to detect social influence between network users. Such foundation is desired to make up the core of a future technical solution. This solution should enable the analyzers of social media to perform their analysis on regular periods with a continuous timeline. Even if these analyzers have little or no technical experience in data processing and visualization

To serve this purpose, we start by determining the common characteristics in available user functionalities on the most popular social media platforms. Then produce a model for data structure based on these similarities in user functionality. By taking a starting point of common user functionalities, we increase the flexibility of this research to be applied to as many social media platforms as possible and perhaps combine results from several platforms in one single analysis if needed.

After establishing an agreement on the data model to be used for collecting and storing crawled data from social media, we dive into the main core functionality of detecting social influence between network users. Multiple techniques of detecting and scoring social influence will be implemented to fit the different needs of a final analysis. The desired result is a user-oriented influence graph, where each node represents a participating user, while each edge between two given users represents the influence between them concerning the direction of influence, and holding the score of influence strength, along with its classified field of influence whether it is in sport, politic, or economy, etc.

Following the previous effort, we evaluate the performance of the influence graph model. Then go through test results from real-life data crawled from a rising social media platform called "Reddit". We will then try to highlight the most interesting and useful features of the produced influence graph and push its power of detecting influencers and their area of influence to the limit. The final two processes of test and evaluation are together a vital step to confidently rely on the quality of the produced influence graph by investigating its informative capabilities in social analysis.

Furthermore, a technical solution is to be designed and implemented to work hand in hand with the theoretical approach. Then function as a possible practical implementation used as a proof of concept and as a helping tool in testing and evaluation. During the process of designing and building this technical solution, many important aspects of data protection, reliability, and availability are discussed and dealt with to improve the ecosystem of this application.

This was a brief introduction to the upcoming research in a nutshell. But first, let's take notes and learn from some interesting pre-attempts in studying the influence between users of online social media.

II. RELATED WORKS

Among the community of data science, a wide variety of studies has focused on extracting information from online social media. A great amount of this effort has been dedicated to studying social influence between users to better understand the behavior of individuals for many purposes. Research of social influence takes different forms and vary in size and scope. Some researchers take on the very fundamentals of detecting social influence, while others dive through it to reveal details such as a specific type of influence, or hidden behavior patterns on different levels.

In the following parts of this section, we explore some related work in the field of social activity on digital media. Trying to get an inspiration that helps direct the effort of this research on the right path.

II - A. Measuring Influence Between Users of Online Social Media.

A good fundamental approach is described by a social network analysis carried out by Y. Guo, J. Cao & W. Lin **[1]**. The fellow researchers divide the influence evaluation models into 2 main categories; the first category is based on network topology which measures social influence between different users by considering the user degree, shortest path, and some random walk characteristics. While the second category bases the influence between users on their interactions through different activities organized in a tree-like structure containing submissions and multilevel comments. However, and despite the reasonably good classification and overview, the published paper of this research lacks some proven results of an experimental approach.

II - B. Data-driven Influence Learning.

A short but rather interesting experimental and mathematical approach is introduced by a paper on Data-driven Influence Learning in Social Networks published by F. Wang, W. Jiang, G. Wang & D. Xie **[2]**. In this paper, the process of influence diffusion is divided into two parts: the launcher (influence strength) and the receiver (influence threshold). This division can generate an accurate and finer-grained influence diffusion model according to this research.

Furthermore, the researchers highlight the importance of having a solid criterium when scoring the strength and threshold properties of detected social influence. Another important acknowledgment is the difficulty and complexity associated with detecting influence relationships between users. This can be a by-product of big datasets that usually include a considerable amount of noisy or less important data points, making it essential for any algorithm used in learning and testing any influence models to perform a minimal scan over the data in the most efficient way possible.

II - C. Alternatives of Information Gathering.

Most well-known providers of social media platforms assist developers and data scientists with instructions on how to crawl their platforms by offering multiple endpoints and methods that can be used for gathering data for analysis.

Multiple researchers shed light on this initial aspect of gathering data from social media platforms. One interesting research mainly describes the alternative of Pushshift Reddit Dataset by J. Baumgartner, S. Zannettou, B. Keegan, M. Squire, and J. Blackburn **[3]**. This research paper offers an undirect but also claimed to be a more efficient and flexible way to gather data from the "Reddit" social media platform, in comparison to using the official Reddit API[1] endpoint.

It also gives an excellent brief description of the FAIR data[2] principles which is highly relevant when choosing the source of data especially when it comes to accessibility and findability.

Another advantage of this research is its extension in discussing a series of the other major alternatives for gathering data from "Reddit", constructively highlighting their strengths and weaknesses.

II - D. Topic Detection in Social Media Platforms.

As mentioned in the introduction, we are set to determine the category of a detected influence between users. This opens for the use of artificial intelligence for classification between different topics, where a certain user activity might fit in. Inspiration on possible solutions for this task can be obtained from the research about annotating and detecting topics in social media forum and modelling the annotation to derive directions. This research is carried out by B. Athira, J. Jones, S. M. Idicula, A. Kulanthaivel, and E. Zhang **[4]**.

A practical case study from an online health community was represented in this research to give a good introduction to data pre-processing and cleaning. Then preceding to construct a reasonable mathematical approach in the training and testing of a machine learning model to be used for topic classification.

Another contribution of this research is the use of various deep learning algorithms to classify posted content such as CNN[3], LSTM[4] , and BiLSTM[5]. All of which enables the researchers to achieve a promising F1-score[6] of about 0.75 to 0.80 in topic classification accuracy.

Furthermore, the above research offers a solution for a much-needed ability to minimize the amount of training data and dealing with the negative effects of label imbalance in a training dataset. Then constructing a convincing conclusion after carrying out a process of well-performed testing and evaluation, where metrics of evaluation are carefully examined and explained in a good scientific approach.

[1] Application Programming Interface.
[2] FAIR data are data that meet principles of findability, accessibility, interoperability, and reusability **[22]**.
[3] Convolutional neural network **[23]**.
[4] Long short-term memory **[24]**.
[5] Bidirectional LSTM **[25]**.
[6] A measure of model's accuracy on a dataset **[26]**.

II - E. Study Case Alternatives of Online Social Media.

It is important to determine which social media platform to crawl under testing and evaluation of a new influence model, to produce a flexible model that benefits the analysis of as many social media platforms as possible. Therefore, it is desirable to work with real-life datasets gathered from a digital media platform that shares common user functionalities with as many popular social media platforms as possible. Examples of such functionalities are submissions, comments, and upvotes or commonly known as likes.

A social media platform that satisfies all these user functionalities is "Reddit" which is examined by the research with the title "Information and Social Analysis" carried out by T. Steinbauer at the University of California, Santa Barbara **[5]**.

Steinbauer starts with a brief but very constructive comparison between the most popular sites for social news with Reddit included. The core of Steinbauer's research lays in his analysis of subreddits, submissions, and comments on the digital platform of Reddit. This analysis help explaining why Reddit should be used in evaluating the performance of an influence model, by viewing the analysis's ability to draw a picture of influence between users of Reddit. The reason for this is Steinbauer's detailed analysis on which subreddits seem to have the most of user's activity.

In addition, Steinbauer constructed an example influence graph between users of Reddit. This graph helps to show which user has the highest influence based on the user's interactions through comments.

However, submission authors are not included in the dataset of the constructed influence graph, making this influence model less reliable if ignoring the often significant role of posters in generating discussions on social media.

Another downside of Steinbauer's modeling of an influence graph is the limitation of not using any other criteria than user interaction through comments, such as the upvote scores or number of thread comments posted on other submissions and comments.

Although Steinbauer has introduced a detailed result overview of his evaluations and analysis, there is still a question mark on the technical details, because algorithms that have been used for producing the model of the influence graph are not provided to the reader in satisfying detail.

III. REDDIT AS A CASE STUDY SOCIAL MEDIA

There exist a wide variety of popular social media platforms and most of them are constantly gaining popularity among users from all over the world.

The following figure shows the market share of the top 7 most popular social media platforms during the last decade from 2010 to 2019. The market share data is obtained from statcounter.com which claims to base its statistics about digital markets on a sample exceeding 10 billion pageviews per month **[6].**

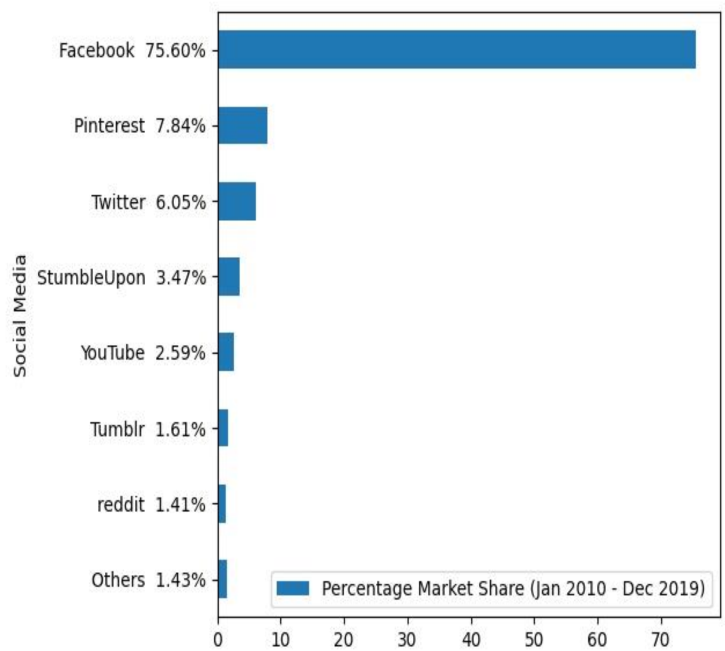

*Figure 1, Worldwide market share of top popular social media platforms from the beginning of 2010 to the end of 2019 **[7]**.*

Although Facebook is the definite leading social media platform, it is still possible to observe a competition in popularity when looking at the less popular social media platforms below Facebook. Reddit has a popularity share of 1.41%, which is nearly equal to all other platforms that are less popular than Reddit, these less popular platforms have a remaining total market share of 1.43% according to the dataset from statcounter.com in **[7]**.

A normal side effect of more popular social media is the large amount of data that users generate on such platforms, which slows the process of extracting data from such platforms. Although data from more popular social media often has higher integrity, it is important to keep a balance between data integrity and ease in findability and accessibility. In this research, we try to compromise between these two factors by choosing a medium-sized social media platform for use during testing and evaluation of the influence graph model.

Reddit is in the 7th rank of popularity above 14 other social media platforms, some of these less popular platforms are well established such as LinkedIn and Instagram. This makes Reddit a suitable candidate to be used as a case study of social media in this research, as Reddit offers our desired moderate balance between network size and easiness to crawl.

In addition, many of the most popular social media platforms tend to specialize in a certain area or field of social activities such as LinkedIn for professional life, and Facebook on the other hand mostly used for private and personal socializing. Some digital platforms combine aspects from both areas such as the so-called digital news platforms, that offer their users an opportunity to interact with each other in many aspects of socializing like professional and personal life combined. Reddit is considered as one of those digital news platforms which are still gaining popularity and increasing in content since its launch in 2005.[7]

A user on Reddit can create or join a group, and post submissions on any group, then comment on any submission or other comments. It is also not obligatory for users to join a group to be active in them or read their content. These groups are called subreddits and tend to specialize in a certain topic of interest in society, and for many users, it is seen in a way that is somehow like reading the newspaper which is often divided into pages for multiple areas of concern such as politic, economy, or sport. The high separation between topics of interest in Reddit makes this platform ideal for testing how well an influence detecting algorithm can discover and classify different types of influence between users.

Reddit differs from other social media platforms in the sense that it attracts users by their interest in topics and happenings in their social surroundings, while other social media often relies on the social affiliation of a future user.

[7] ***Reddit is a social news website and forum where content is socially curated and promoted by site members through voting. The site name is a play on the words "I read it."*** **[27]**.

Moreover, many other social media platforms share a lot of common user functionalities with Reddit, such as groups, submissions, and comments.

This high similarity between Reddit and most popular social media platforms, along with Reddit's ability to separate users into multiple different social groups, makes Reddit very suitable as an evaluation study case for this research because common functionality increases the modeling flexibility to be used on other social media platforms in future analysis, and Reddit's separation of social environments in subreddits serves the purpose of comparing the predicted type of social influence between users to the actual definition of the group where the interaction between users has occurred. This gives us an idea of how well our influence model is classifying topics of social influence.

Although Reddit is a user-oriented platform, its users often prefer to be anonymous, which is useful when presenting results without having to worry about neutrality issues, but Reddit's users do also have the opportunity to identify themself personally if they desire by using their username.

Another good reason for choosing Reddit as a study case is its highly developed endpoint crawling API, which is very object-oriented and offers an advanced wrapper library for the Python language. This eliminates the bother of dealing with HTTP requests and latency issues, as all of this is taken care of in the background of the Python Reddit API Wrapper. The Wrapper is free to use but it requires a registration, which once obtained offers no restrictions on how often Reddit is crawled, unlike crawling by adding ".json" to the URL[8] which has its downsides such as the limitation for under 100 submissions at a time, and the blockage of multiple requests from the same IP address as a prevention measure from Reddit to stop denial of service attacks. All these downsides are escaped by using the Python Reddit API Wrapper which increases the reliability and stability of data streams from Reddit. As a result, the PRAW[9] python module satisfies the following FAIR data principles:

- **Findability:**

  Once using PRAW, it is easy to find and retrieve data from Reddit no matter how detailed the data is.

[8] Uniform Resource Locator, commonly known as a weblink
[9] Python Reddit API Wrapper

- **Accessibility:**

  As mentioned earlier, a programmer does not have to deal with HTTP requests and latency issues, because using a traditional API endpoint makes the programming experience much easier allowing programmers to focus on the objective of their work.

- **Interoperability:**

  A good history of documentation and maintenance of the PRAW module **[8]**, along with its popularity between programmers gives it an excellent record of ability to integrate with different products and systems that need to communicate with Reddit.

- **Reusability:**

  PRAW is highly object-oriented in both query language and retrieval results. This is helpful for the usability for integration in different projects and technical solutions both in the present and future technologies.

A complete analysis of the current and future potential of Reddit is performed in a master thesis with the title "Analysis of User Attention on Reddit" by Elias Zeitfogel at the Graz University of Technology **[21]**. This thesis gives an excellent analytical review of many features and other segments of Reddit. And it can be used to establish a solid understanding of Reddit's capabilities as an online social media platform.

Based on the above four FAIR data principles and the previous analysis of user habits and possibilities on Reddit compared to other social media platforms, Reddit makes a good case study in the testing and evaluation process when we are seeking to detect user influence and their area of influence.

However, to make our research easy to apply on other social media platforms, we shall design our ground data structure to adapt for the common user functionalities between Reddit and the most popular social media platforms. This step is elaborated in the upcoming section on the ground data structure.

IV. DEFINING A GROUND DATA STRUCTURE

The flexibility of design is an important requirement of this research because we aim for a future technical application that uses the influence modeling developed in this research on as many other social media platforms as possible.

Although this might be difficult to achieve because of a wide variety of available social media platforms and their different user functionalities. It is still possible to observe some common user functionalities between the most popular social media platforms such as LinkedIn, Facebook, and Reddit. This common functionality is no accident because social media platforms are most likely inspired by real-life social interaction to begin with, which in turn is a natural advantage for our application.

After studying the available user functionalities in Reddit compared with these same functionalities on the most popular social media platforms, it is easy to see a big potential for developing a generalized data model that can be used to structure data crawled from any of the applying social media platforms. It is therefore important to consider the desired results of this research before establishing a ground model of data structure.

Social influence can be defined to be the ability of one society member to change the thoughts or behavior of another society member, and although this definition is simple, the complexity is hidden in the way social influence plays out in societies. Some people get influenced without showing any significant reaction that can be recorded and studied, such influence is said to be of a passive type of influence, an example of passive influence is reading a newspaper where the reader gets influenced without adding any additional comments to the content.

The main goal of this research is to use recorded data from social media to visualize the influence flow between a group of people in their social interaction. For this reason, we are going to look at active social influence where we would expect the person who gets influenced to react by submitting an activity on certain content. This requires an activity-based model, where activities such as submissions and comments are considered as indicators for social influence.

The second requirement of this research is the importance of visualizing the flow and direction of influence between members in a social media interaction. For this purpose, we will be building an interaction-based model that can retain the origin and target of each detected and measured influence, which benefits the storage of influence direction, and in the big picture can be used to visualize the entire flow of social influence between society members.

The model of the ground data structure of this research can initially be based on four different entities a user can create and interact to; these entities are:

1. **Network**

   Holds information about the crawled social media platform(s) or a smaller segment of it.

   This entity can also refer to multiple segments from different social media platforms if there is a need to study multiple social media platforms at the same time.

2. **Group**

   A group contains a bundle of submissions posted by users of the group. It also contains information about a certain group in form of identification and other attributes such as the group ID and name.

3. **Submission**

   A submission is posted by one user and is assumed to be in text format with the possibility of further extension to multiple other formats like images and links in future improvements. The submission entity has multiple useful information stored in its attributes such as the current number of comments and upvotes on this submission, along with information about the author of this submission and other identity attributes.

4. **Comment**

   The comment entity is very similar to the submission entity containing a body text, identification of the author, and information about the location of a comment in the comment thread. And therefore, it has an additional feature which is its ability to be a parent and/or a child of other comments and submissions. This means that comments can be modeled as a tree data structure that can grow endlessly.

Figure 2 below shows a diagram of an entity-relationship model that will make the base of our data structure further on in this research. In addition to the four established entities, four relationships bind these entities together defining their relations to each other. A network can contain multiple groups, and a group can contain multiple submissions or posts, where users can either comment on those submissions or on other comments which are child descendants in the comment tree of a certain submission.

Most of the popular social media platforms contain the four identified entities in this ER-model[10], and although they might have a different name, form, or purposes such as a company page on LinkedIn or a user profile on Facebook, both can be treated as groups just like Reddit groups as well.

The attributes of entities open for more flexibility as we might need to shrink or extend our ER-model in the future, an example here is by not including the upvotes attribute, or by adding a reaction attribute to submissions and comments. But in this research, attributes are generalized to match the very common details about their entities on the most popular social media platforms including Reddit.

Now that we have a ground Entity-Relationship model to base our data structure on, we can proceed into discovering the influence between platform users based on the interactions between them by using these entities and their relationships established in this section.

In the next steps, we will elaborate on the different stages of our influence modelling process for detecting and extracting social influence from online social media. The algorithms explained in the upcoming section of influence graph modeling are expected to have multiple dimensions for revealing the strength and types of social influence between different users of online social media.

[10] Entity-Relationship model

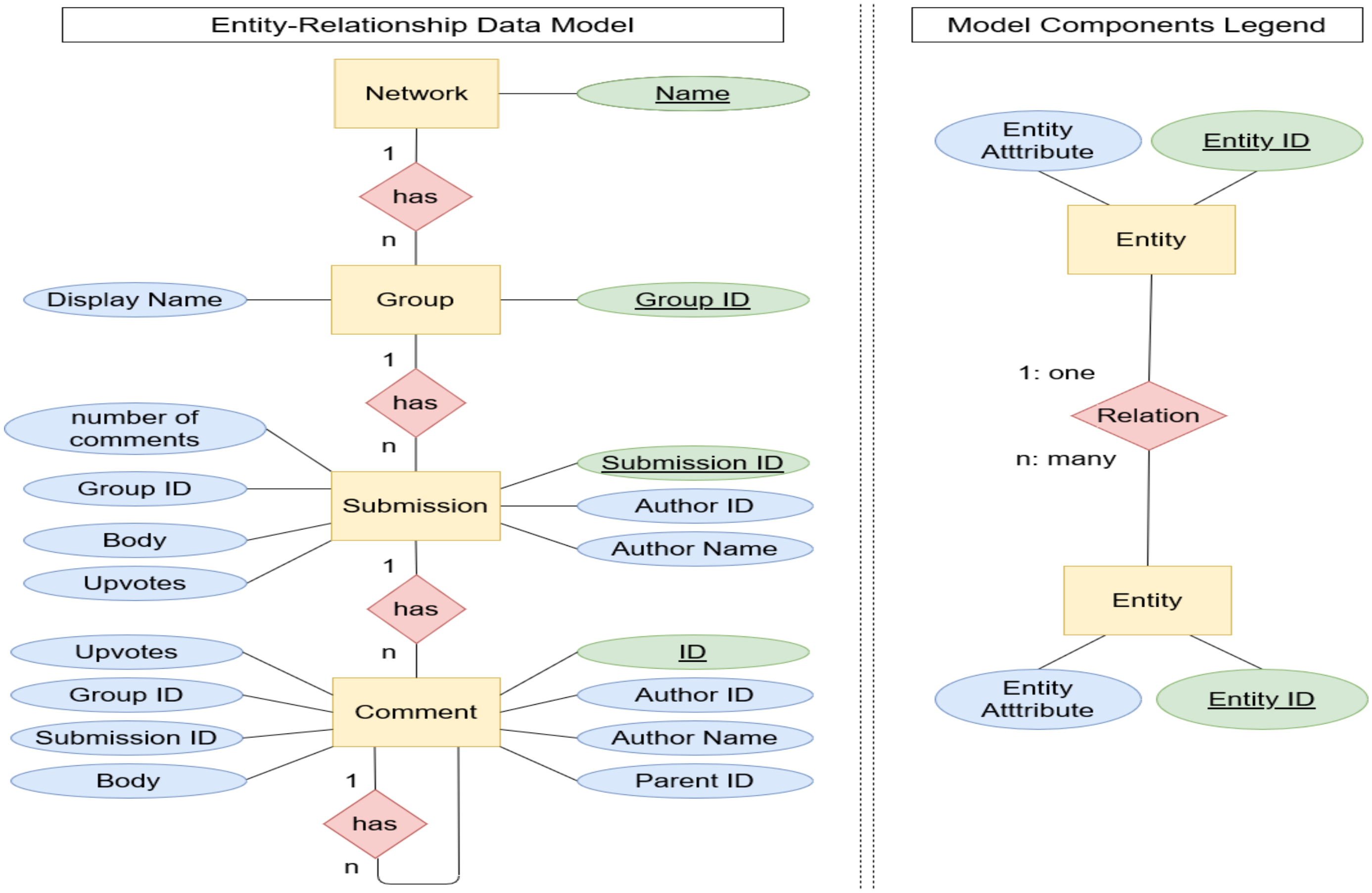

Figure 2, The Entity-Relationship model represents the ground data structure of this research.

V. Influence Graph Modelling

V - A. The Activity Thread.

The previously established model of the ground data structure in figure 2 provides the required entities for detecting social influence based on activities and interaction between users. We can observe a clear hierarchy between the following four entities: network, group, submission, and comment. This hierarchy enables us to model the data from a social network in a tree structure, where the network contains multiple groups that in turn contain a series of submissions, which can contain multiple branches of comments. Comments can also have their underlying comments resulting in a tree that can grow endlessly as users add more comments on previous comments.

Figure 3 below shows a small example of an activity thread tree of just one submission, which contains the activity of the submission in the root position and its several branches of comments each in its respective hierarchy level. We can call this thread the activity thread as every single node in this thread represents an activity object that a certain user had created.

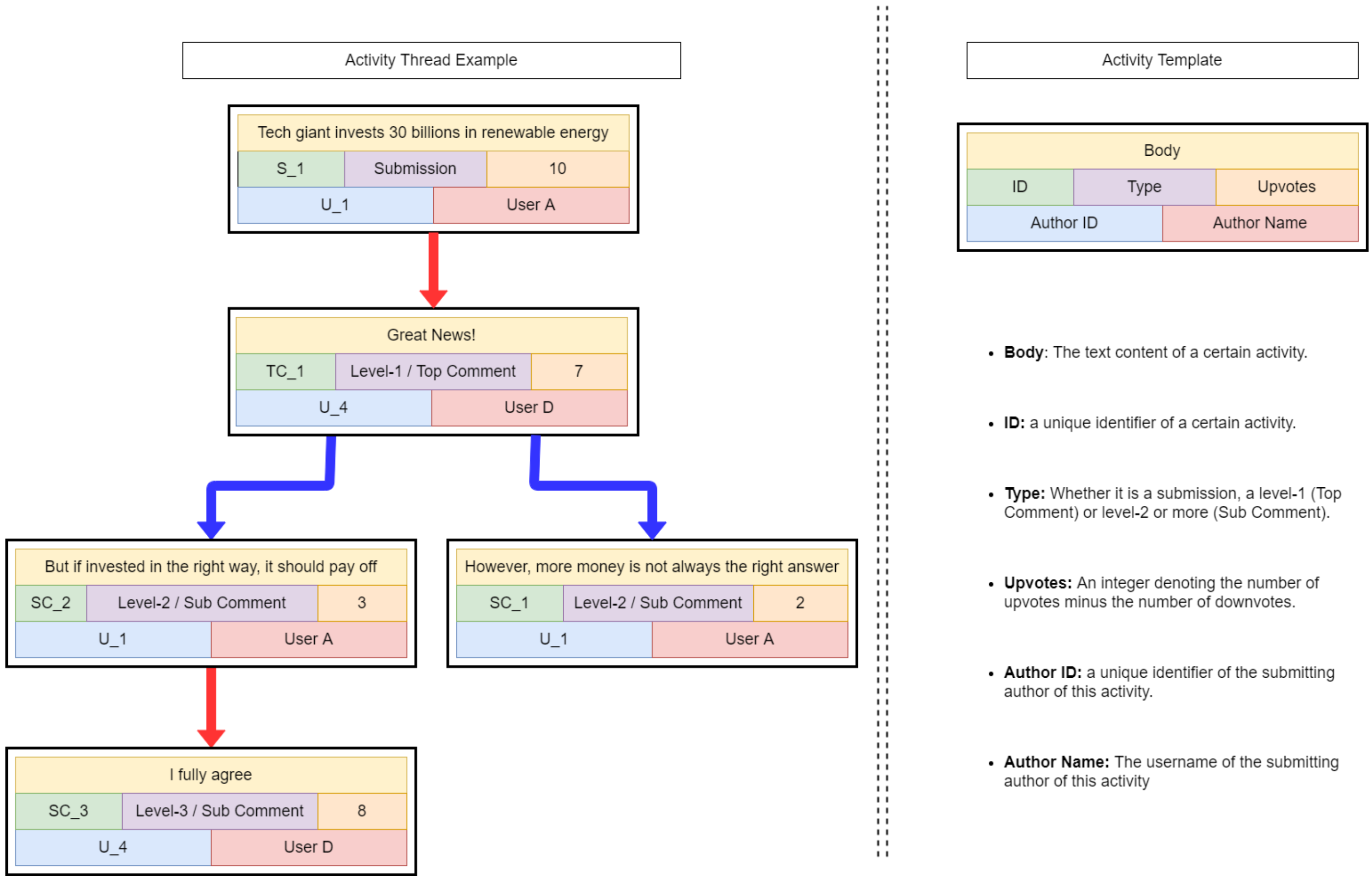

Figure 3, A set of submissions and comments can be modeled as an activity tree by taking advantage of its natural thread hierarchy.

In the activity thread drawn in figure 3 above, we have 2 participating authors. The directed arrows indicate that the target activity was an interaction on the source activity. Where a red arrow indicates an interaction from "User D" to the activities of "User A", and a blue arrow indicates an interaction from "User A" to the activities of "User D". In the next steps of influence modeling, we will investigate how we can use this activity thread to give birth to a weighted and directed activity-oriented graph that shows the interaction flow between users performing these activities, along with other details such as influence type and score magnitude.

Before we head further into more details, we should control the endless growth of the activity thread by defining comments based on their level in the activity tree. This is to avoid any confusion when referring to different types of comments. Because comments can have their underlying thread comments, we can establish a categorization of them based on their hierarchal level in the activity thread tree, where we can use the following two types of comments:

- **Top-Level Comments**

  Which are comments made directly on a submission.

- **Sub-Level Comments**

  Which are comments made on top-level comments or other sub-level comments.

The distribution of comments between top comments and sub comments can give us a picture of how involved members are in each submission because observing more sub comments than top comments can indicate that authors are taking their time to read and react to top comments in addition to the submission. Which is an indication of a greater engagement from authors than if they just keep themselves to writing top-level comments.

V - B. Scoring Influence.

Since every tree is naturally a graph, we can take benefit from our hierarchical activity thread to construct a directed graph, where each edge in this graph is directed from a parent activity to one of its child activities, and representing one interaction between 2 network users, where the target and source nodes are represented by their authors. This shows that the author of the target activity has reacted to an activity submitted by the source author, which in turn can be considered as an influence indication directed from the source author towards the author of the child or target activity.

Knowing the direction of influence helps us detect influence between users, but it is not satisfying enough to give an idea of how strong a certain influence is. Therefore, we aim to grade all detected influence between network users, so we can determine how strong a certain influence is in comparison to other influence edges in the influence graph topology. For this reason, we will assign a strength score to each influence edge between two persons.

Many measures can be used to perform the scoring of influence. However, each scoring measure has its strengths and weaknesses. One way to reinforce the analysis system is by offering multiple scoring measures to be used in the analysis. To accomplish this, the following three different scoring measures are used in this research, each having some strengths and weaknesses depending on the use case of the final influence analysis.

1. Interaction

This scoring measure counts how many times a studied user has reacted to activities performed by another user. The strength of this measure is in its ability to detect each interaction between users and differentiate them using the number of interactions as a score.

It can be observed that many users who are interested in a certain topic tend to quickly give up on submitting comments and take on the role of watchers. To accommodate such cases of low interaction, the following two scoring measures can be applied in the analysis.

2. Upvotes

By using the difference between upvotes or likes and downvotes or dislikes on a parent activity, we rely on the audience's opinion of the parent activity given by other users. The submitting author of this activity is the influencer, and its influence takes the score calculated as the difference between upvotes and downvotes, this influence is then directed towards those users who comment on this parent activity, and thus they get influenced by the author of this activity having this influence score.

This scoring method gives a democratic approach that enables us to know whether an influencing activity is supported or downvoted by a group of an interested audiences. At the same time, it is important to notice that in some networks such as Facebook, users can't downvote an activity, which leaves us without knowing for sure whether the audience is most likely to support an activity or not. However, it is still possible to accommodate this by counting the number of upvotes as a measure where 0 is the lowest score of an influence.

A possible downside of this upvotes scoring measure is that all child activities will get the same score of influence, which can result in a low differentiating effect between influence edges in the influence graph.

3. Activity

The activity influence measure helps rate influence based on how many activities descend from a certain parent activity in the activity graph.

Central people in society like politicians tend to influence most people by triggering a great number of cascading response activities. This scoring measure is based on the general extended impact of an influencer and can therefore help identify the central influencers and which users do they have an influence on.

The upside of this activity-based scoring measure is its ability to include the impact of all activities in an influence branch and still be able to differentiate between influence edges on child activities because of its branch-oriented calculation, as it does not take into account all activity branches of a parent activity.

Figure 4 below contains a graph built on the skeleton of the previous activity thread in figure 3. It converts nodes from representing an activity object to represent the author of activity while preserving the hierarchical structure as it is in edges and their directions. It also introduces the three scores of interaction, upvotes, and activity as separate weights calculated for each influence edge.

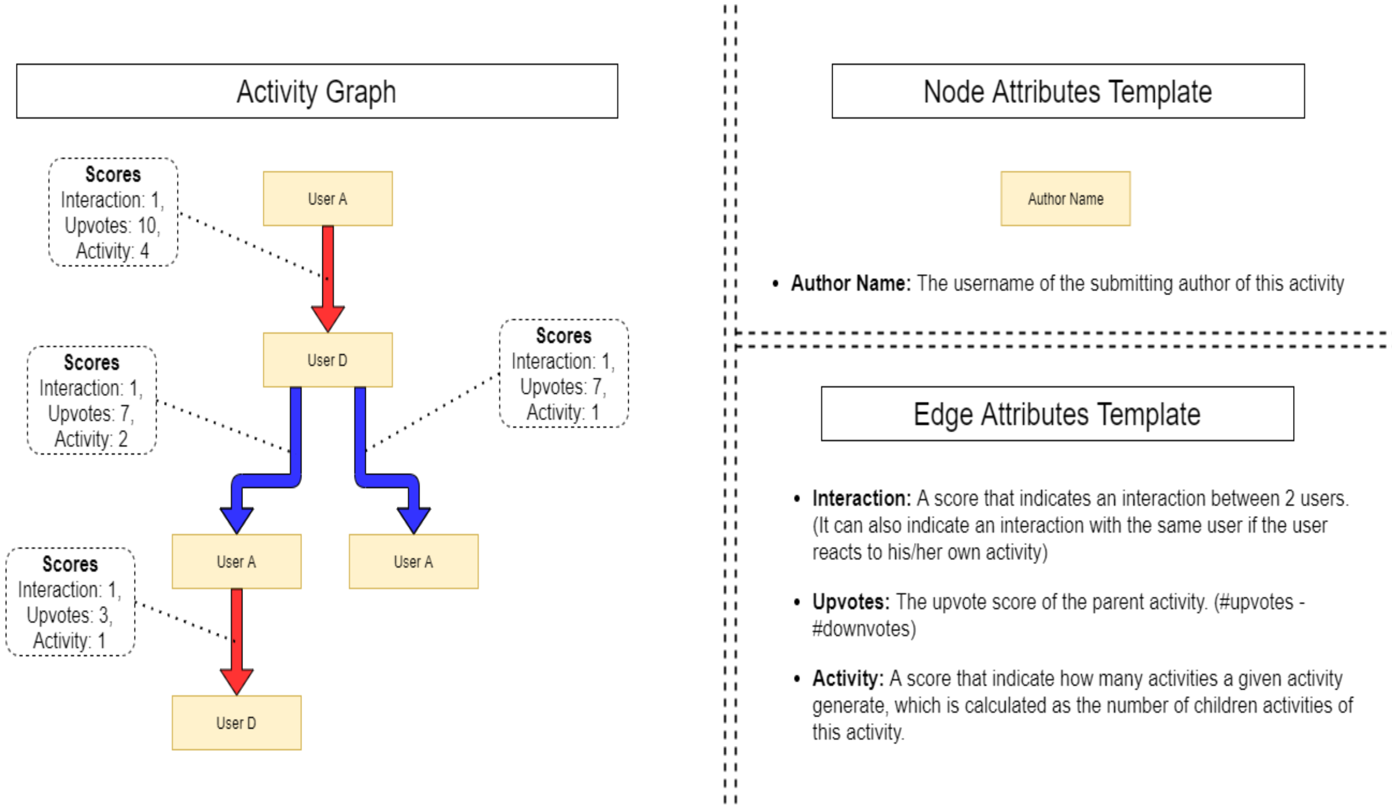

Figure 4, The activity graph is constructed by representing the nodes of activities using their authors and applying the scoring techniques of interaction, upvotes, and activity to score the individual influence edges.

When examining the activity graph example in figure 4, we can count 4 influence edges, half of these are directed from "User A" to "User D" and the others two are oriented in the opposite direction. We can digest the structure of this directed and weighted activity graph by merging all nodes and influence edges having the same unique pair of source and target authors of activities. While performing a simple addition of the respective scores for each of the three scoring measures. The final produced graph is to be called the influence graph with more details and examples on how to construct this graph model presented in the next section.

V - C. The Influence Graph.

After going through the activity thread tree and its transformation to a directed and weighted activity graph, it is now easier to digest such a graph to produce the desired output of an influence graph between network users, where each node represents a unique author in this graph, and edges are used to indicate a particular scored influence between two persons concerning the direction of influence.

Looking at the previous activity thread in figure 3 and the activity graph in figure 4, we can count 2 submitters who are "User A" and "User D", and together they submit 5 nodes of activity objects. To find out how much influence does "User A" have on "User D" and vice versa, we go through the following four steps:

Step - 1: Creating a person node for each unique author in the activity graph.

Step – 2: Listing all influence edges stretched from "User A" to "User D" in the activity graph.

Step – 3: Merging all influence edges in step 2 by summing the values in each scoring measure to produce the respective score of each measure for this influence edge.

Step – 4: Step 2 and 3 above is to be repeated to find the influence in the direction from "User D" to "User A", and the same goes for finding influence between any possible pair of two persons in the graph.

To further clarify the merging calculations of influence scores, we notice that an interaction score from "User A" to "User D" would be the number of edges going from activities submitted by "User A" and interacted to from activities submitted by "User D". Likewise, the upvotes score is the sum of upvotes recorded on the corresponding edges in the activity graph before merging and the same goes for the activity score as well.

Following the four steps above, a new influence graph is born with a unique node for each person, and with its influence edges having three scoring attributes that measure the strength of each represented influence. This graph is now able to visualize the flow of influence between persons, how strong each influence is, and in which direction between them it is observed. A visualized example of this influence graph is presented in figure 5 below based on the previous four steps transformation from the activity graph in figure 4.

This influence graph below does have two network users with two influence edges between them; according to the interaction score, both users does have equal influence on each other, while using the upvotes scoring technique tells us that "User D" has a little more influence on "User A", and the opposite is observed when using the activity scoring measure.

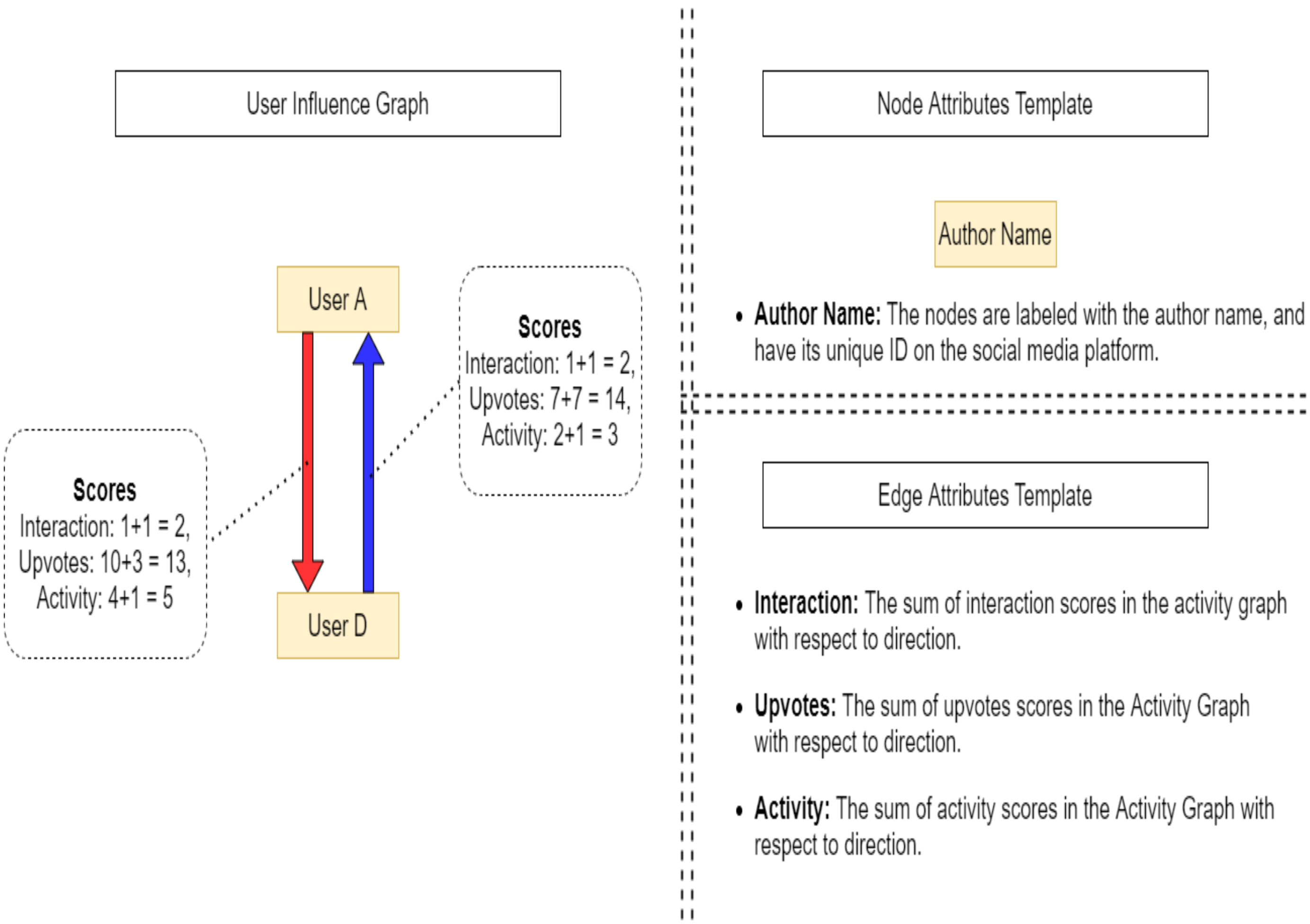

Figure 5, A small example of an influence graph that contains a unique node for each person, where the edges of this graph indicate the weights and directions of influence between network users. This influence graph is built based on the merging of edges from the activity graph in figure 4.

V - D. Testing & Evaluation of The Influence Scoring Techniques.

So far, we have proposed a staged method that is meant to detect influence between users who submits various activities such as submissions and comments on social media platforms. The output of this method is an influence graph where each node represents a person, and each edge indicates an influence from its source person into its target which is the influenced person.

We are primarily interested in the overall capability of the influence graph to visualize the flow of influence between activity authors in a dataset. For this reason, we are going to focus on the edges of influence between users as they hold the score values obtained by using different influence scoring measures that were detailed previously in this research.

We hereby plan to test the score distribution in the edges of the influence graph after observing the effects of applying each scoring measure (i.e., Interaction, Activity, and Upvotes) to a dataset crawled from Reddit, and contains the top 3 newest submissions taken from each of the top 3 most popular subreddits according to internal Reddit statistics accessible by using the python Reddit API wrapper on the 30th of July 2021, 12 PM.

The following figure shows 3 box plots and 3 histograms that visualize the distribution of score values of each scoring measure recorded in the edges of the influence graph.

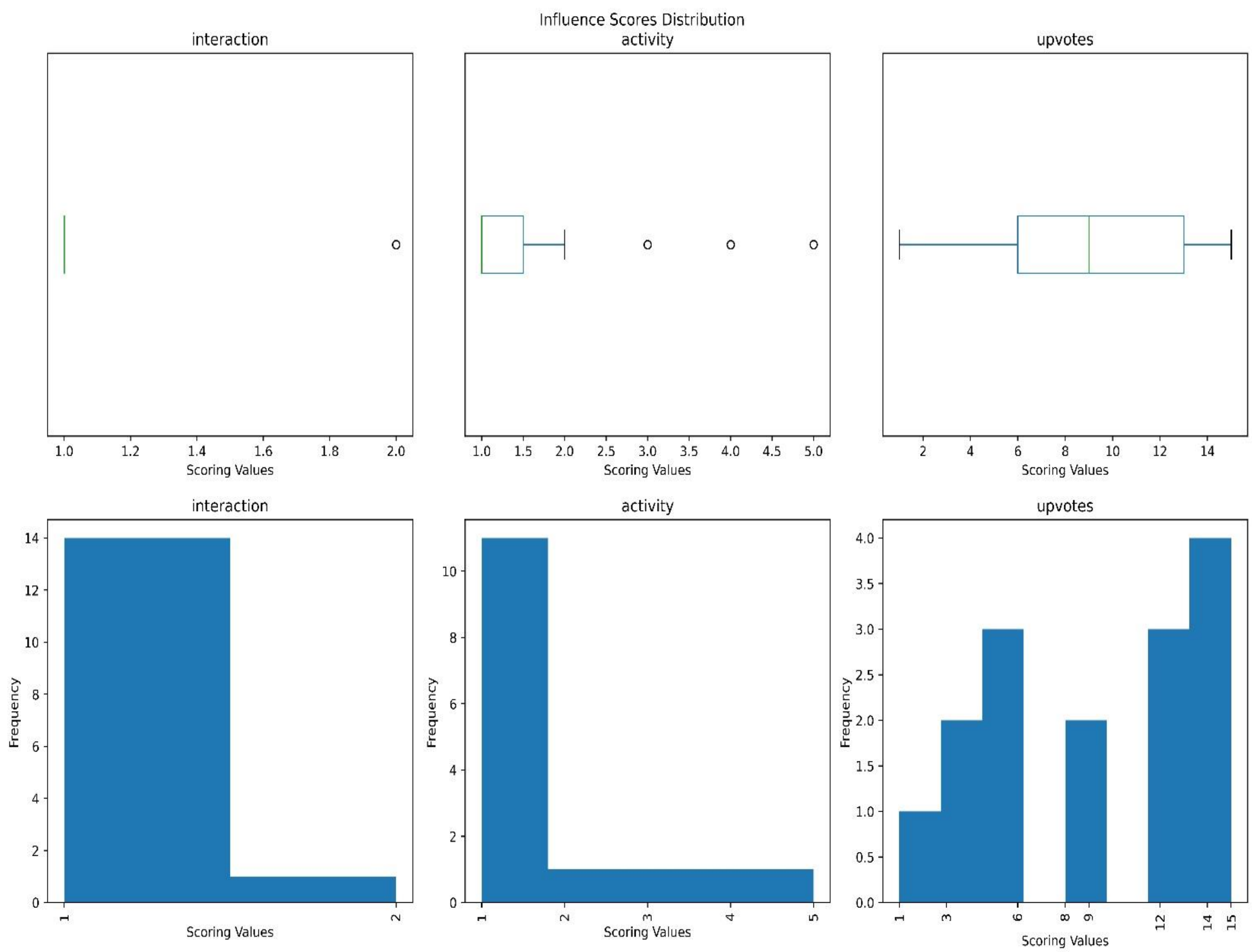

Figure 6, The single score distributions recorded in the edges of the influence graph, constructed from a dataset containing the top 3 newest submissions, posted inside the top 3 most popular subreddits on the 30th of July 2021, 12 PM.

Examining the distribution of the interaction score values in the left box and histogram plots in figure 6 above, gives a clear observation of low variance between score values which reduces the differentiating capability of the interaction scoring measure. However, a relatively good number of 15 influence edges were successfully detected from the crawled dataset.

Moving on to the examination of activity score values in the middle of figure 6, we notice a slight decrease in the number of influence edges having the lowest score in comparison with the interaction score values on the left side.

In addition, there is a noticeable small increase in variance as the range of recorded score values has increased from 1 to 2 in the distribution of interaction scores to a range between 1 to 5 using the activity score values.

Three outliers on the right side of the box plot of activity scores also tell us that this scoring measure has more potential to differentiate between different strengths of influence and help present those detected influence edges that are having more significance than others in the influence graph.

Finally, we examine the upvotes score values to discover a higher achieved variance and no outliers in the distribution of obtained score values. This result shows us the effectiveness of using the upvotes scoring measure for differentiating between detected influence edges.

Because each scoring measure might have its strengths and weaknesses according to the purpose and motivation of the performed analysis, an analyzer might also wish to combine multiple scoring measures and take a look at the distribution outcome of score values to have a better understanding of the influence model.

In this research, the single scoring measures are combined by performing an addition operation from all possible combinations of the 3 scoring measures in each edge in the influence graph. This results in the following four new scoring measures:

- Interaction + Activity
- Activity + Upvotes
- Interaction + Upvotes
- Total = Interaction + Activity + Upvotes

Figure 7 below plots the distribution of score values obtained from the paired scoring techniques. A wider distribution of score values is observed on all the combination pairs of scoring measures compared to the single scores in figure 6.

The combined scores of “activities plus upvotes” seem to give the highest achieved variance, because of its wider spread of scoring values and lower score frequencies. Therefore, the “activities plus upvotes” scoring measure has the best capability of differentiating between detected influence edges based on their scores. But depending on the purpose and motivation of analysis, other scoring measures with less spreading distribution along the x-axis might be more useful.

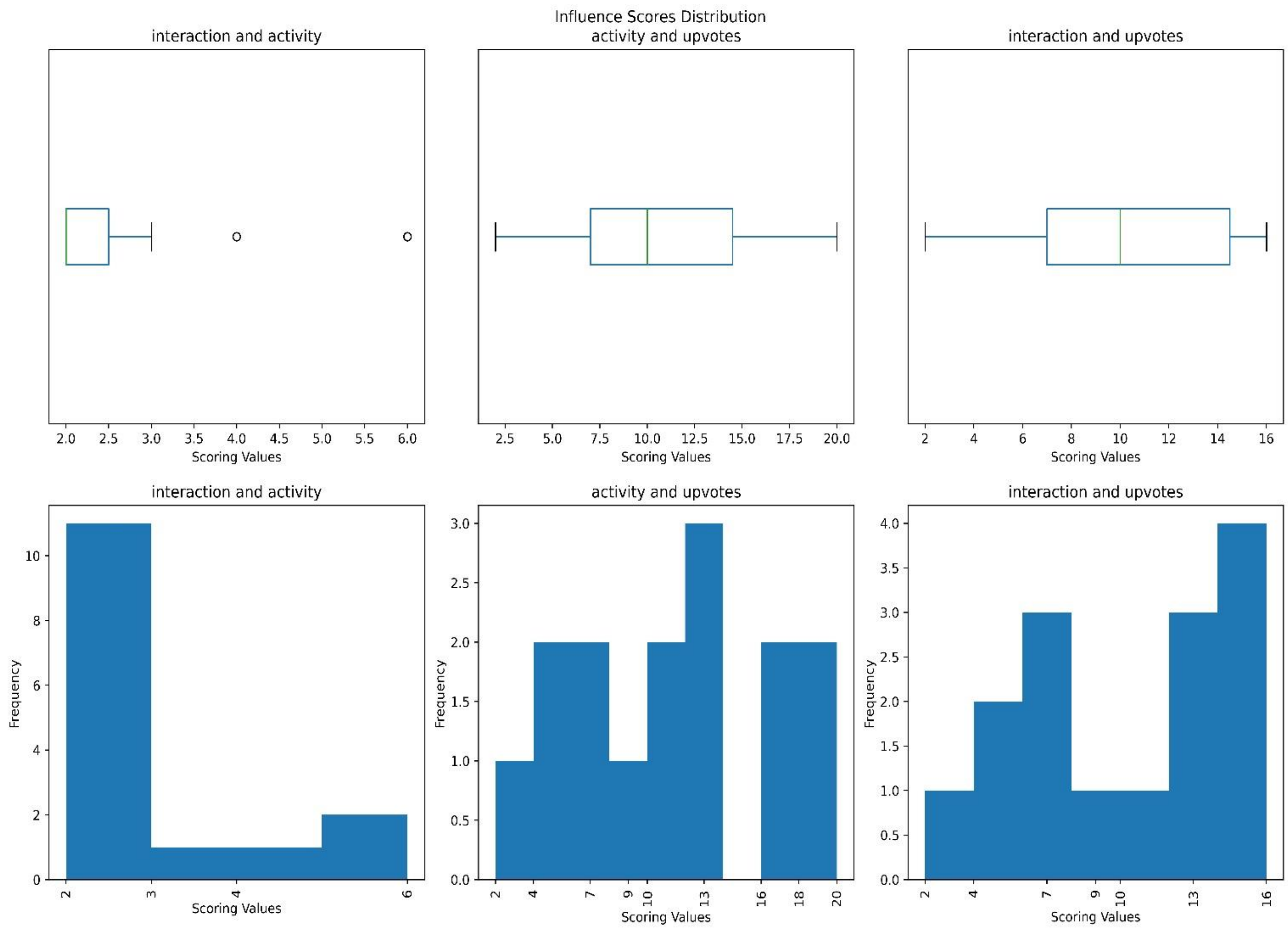

Figure 7, The double combined score distributions recorded in the edges of the influence graph, constructed from a dataset containing the top 3 newest submissions, posted inside the top 3 most popular subreddits on the 30th of July 2021, 12 PM.

Furthermore, it is also possible to examine the results of combining all single scoring measures to produce a total score, where the distribution of its scoring values is plotted in figure 8 below. This total scoring measure gives us an even wider range of possible score values, but with nearly as equal variance as the combination of the "activity and upvotes" scoring measure, reasonably because of the low differentiating capability and low variance of the added interaction score values.

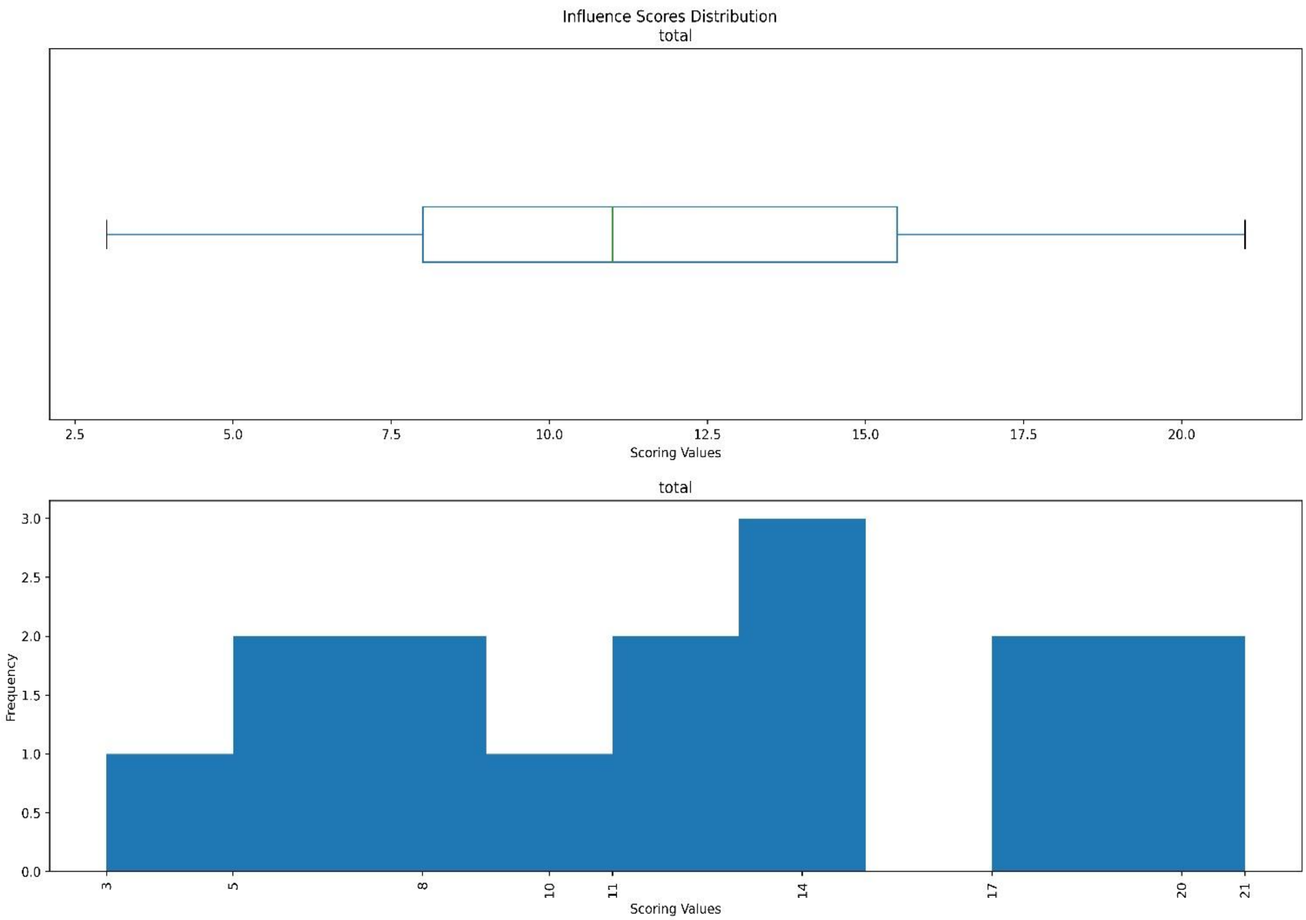

Figure 8, The total score distribution recorded in the edges of the influence graph, constructed from a dataset containing the top 3 newest submissions, posted inside the top 3 most popular subreddits on the 30th of July 2021, 12 PM.

Based on the previous testing results, we can conclude that each scoring measure is mostly dependent on the nature of raw data generated by users on a social media platform, as network users may have different ways of using the social media platform and therefore different ways of influencing each other. However, it is possible to mitigate the effect of how users use the network to influence each other by using the combined influence scoring measures, which can have either an advanced formula or be as simple as an addition of the single scores in each influence edge of the produced influence graph.

V - E. Influence Field Classification.

Being able to identify influencers on a social media platform is one objective of this research, another objective is to classify detected influence in multiple distinct topics or influence fields such as politics, sport, or economy. Knowing the type of influence between people enriches the process of analysis and gives it a topic-specific dimension that helps the analyzers to focus on the topics of interest during influence analysis.

In this section, a classification method is proposed by integrating a text classifying machine learning model into the process of building the activity and influence graphs.

Due to the normally large amount of detected influence between people on online social media, a manual classification between different fields of influence is hard to perform and proves its reliability as topics can be perceived differently from one person to another. This is where artificial intelligence can come in handy by building a machine learning model with the task of classifying influence into multiple topics learned from a training dataset.

This training dataset can be static over time, or dynamic and continuously changing. The advantage that can be gained by having a dynamic training dataset is its ability to stay updated with the newest details that differ in different fields of influence. While a static training dataset will be frozen in time and might give false classifications in future use, due to its outdated dataset.

For this reason, a dynamic process of updating the training dataset is implemented in this research. And is used in the training of a text classifier. More details are included in some of the upcoming technical sections about the process of crawling periodic training datasets and building the text classifier used to determine the field of influence based on texts from submissions and comments.

For the simplicity of this section, we assume that we have a trained text classifier ready to be used in the classification of different fields of influence based on an input text.

When transforming the activity thread into an activity graph, we classify the influence type of each edge in the activity graph by constructing a combined text from the submission activity at the root of the activity thread, along with the texts from the source and target activities of an edge. Then input this constructed text into the available text classifier to have an estimate about which influence field a particular influence edge in the activity graph is most likely to belong in.

Adding the text of the submission activity to each classification trial increases the chances of having a more correct classification because the submission text tends to have a clear indication of the topic of discussion in its comment thread. However, using only the submission text as an input for the classification of each activity edge leads to a little informative classification as all edges between activities in the activity thread will have the same classification. Therefore, the texts from the source and target activities of an influence edge are also added to the input of the classification trial. This helps detect any derailment between topics under the discussions in the multiple threads of comments.

An overview of this process is visualized in figure 9 below, where 4 classifications had been performed on the 4 edges in the activity thread shown on the left of this figure. Then, we label the corresponding 4 edges in the activity graph with the respective classification output. This is shown in the middle graph in figure 9.

An example of a classification process is the edge between the comment “TC_1” and “SC_1” in the activity thread, here we construct an input text from the following activities:

- Submission activity: “Tech giant invests 30 billions in renewable energy”
- Parent/Source activity: “Great News!”
- Child/Target activity: “However, more money is not always the right answer”.

After adding spaces between these 3 texts, the classification method of a trained text classifier is used, which we assume to give us an influence classification in “Technology”. This classification is stored in the edge between the activities “TC_1” and “SC_1” and shown in its respective edge between users in the activity graph at the middle of figure 9.

Also, note that the text classifier might give a non-expected or false classification as simulated in figure 9 between the activities of “TC_1” and “SC_2”, or between “SC_2” and “SC_3” where we would expect a classification in technology or economy.

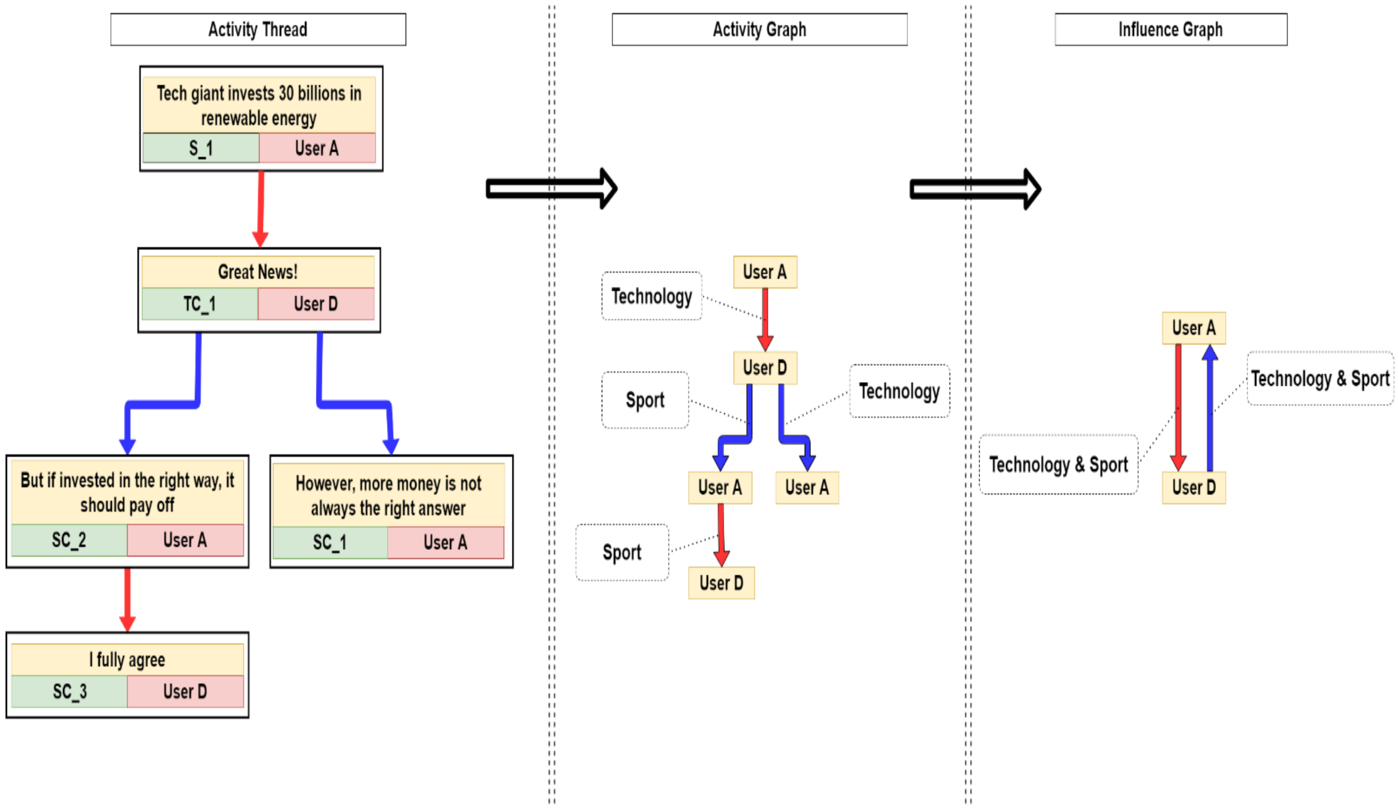

Figure 9, An example overview showing the process of influence field detection, which is an integrated process in the skeleton of building an activity graph and its final transformation to an influence graph.

Finally, we gather the distinct classified fields of influence in each respective edge in the influence graph during the merging process from an activity graph to an influence graph. Noticing that an edge in the activity graph can only have one classified field of influence, but when constructing the influence graph, multiple edges that are having the same combination of source and target persons would be merged, and as a result, all distinct classified fields of influence will accumulate in the respective merged edge in the final influence graph.

When examining the texts submitted by authors in the activity thread of figure 9, we might want to manually conclude that the influence constructed from this thread is to be reasonably related to technology. And although the assumed text classifier might falsely classify some influence in the sport category as simulated in figure 9, a final look at the influence graph confirms that the technology category is also included in the respective edges in the influence graph. This tells us that our implementation of influence field detection can investigate and capture any derailments of discussions under the activity thread, which represents a strength that enables this method to look beyond what the human eye can assume about written discussions.

This classification process is very similar to the process of influence detection, as it has the same three staged skeleton, which plays out by transforming the activity thread tree into an activity graph, then into a final influence graph.

This similarity makes it easier to merge the three processes of influence detection, influence scoring, and influence field detection into one algorithm, which can have the effect of reducing the runtime of a practical algorithm that implements these three processes all at once. Therefore, the need for multiple iterations over records in the dataset is minimized. However, this can have a moderate increase in the complexity of such implementation, but this complexity can be dealt with using the recommended best practices of programming.

V - F. Testing of Influence Field Classification.

Performing a reliable process of text classification is the core challenge in detecting different types of social influence, as there is a need to have a reliable well tested, and tuned text classifier for more accurate helping in classifying between influence fields.

In this research, we will be using a well-documented text classifier inspired by the developer community on "scikit-learn.org" **[9]**. This classifier is a highly optimized ML[11] model that combines the use of multiple ML algorithms for text classification.

The text classifier has a pipeline architecture that starts with text vectorization which maps each unique n-gram[12] term in the provided training records to its occurrence count. Followed by a transformation process that can benefit from using the measure of Term Frequency-Inverse Document Frequency **[11]**. The last part of this pipeline is a model of text classification that is to be built on top of vectorization and transformation to produce a text classifier used to categorize an input text into multiple topics learned from the provided training dataset.

The scientific field of text data analysis is very diverse and offers multiple techniques for processing and extracting features from text data. The technical implementation of text data analysis in this research was generally guided from Part III of the book titled "Text data management and analysis: a practical introduction to information retrieval and text mining" by C.X. Zhai and S. Massung **[15]**.

To solve the challenge of providing training records, it is possible to have a training dataset made up of categorized text which enables supervised learning and allows to estimate the topic of a new unseen text.

As with most supervised ML models, the model's ability to classify text with high accuracy is highly affected by the quality of its training dataset, and the different key parameters for the algorithms used in building the model of a text classifier. In the following steps, we will describe the process of gathering training data, testing its initial quality, and tuning the text classifier key parameters for optimal classification.

- **Step – 1: Gathering training data**

We start by gathering training data from different subreddits, each is having a matching name to one of the target categories that we are interested in under analysis, an example of a category collection can include politics, economy, sport, entertainment, and technology.

The top 125 newest submissions on these subreddits are crawled from Reddit if available, meaning that we can have up to 125 text records from each subreddit, and every group of 3 subreddits can be trusted as a text provider for a certain category of influence fields, we then label text from these subreddits with the target category name as shown in table 1 below.

Following the crawling plan in table 1, we obtain up to 1 875[13] training records, these records would then be fed to the text classifier that is used for influence field classification during the process of building the influence graph.

This way of gathering training data from the same social media platform makes the text classification model up to date in a continuous timeline. Training datasets can then be obtained at periodic intervals and provide the text classifier with new categorized text submissions that reveal the latest events happening in the different categories.

The text classifier is now able to stay updated with the latest categorization of topics according to the active community and not in a fixed dataset. However, the downsides of this method are the need to constantly consume system and network resources to crawl new training datasets. In addition, having to tune and evaluate the performance of the newly built text classifier after each updating replacement of periodic training datasets. These downsides can be mitigated by keeping a moderate size of the training datasets to enable quick data crawling and tuning of the text classifier, which was a suitable solution in the technical part of this research after following the crawling plan in table 1.

[11] Machine Learning.

[12] ***"An n-gram is a contiguous sequence of words in a text."* [10]**

[13] 1 875 = (5 categories) X (3 subreddits for each category) X (125 crawled submissions from each subreddit).

Table 1, A plan for gathering a dynamic training dataset using selected subreddits from Reddit.

Category	Subreddits	Number of training records (Up to)	Example record
Politic	politics	125	title: "elections is postponed due to security reasons", label: "Politic"
	PoliticsPeopleTwitter	125	
	elections	125	
Economy	Economics	125	title: "The price of oil is at record low", label: "economy"
	economy	125	
	business	125	
Sport	sports	125	title: "3 days until the kickoff of FIFA world cup", label: "sport"
	olympics	125	
	worldcup	125	
Entertainment	movies	125	title: "show to be canceled due to bad weather conditions", label: "Entertainment"
	comedy	125	
	culture	125	
Technology	technology	125	title: "new material used in batteries might revolutionize electric cars", label: "Technology"
	science	125	
	Futurology	125	
Total		Up to 1 875 records if available	

Before starting to evaluate and tune our text classifier, we shuffle the data randomly to guarantee a fair distribution of categories throughout the dataset, then freeze it by using a constant seed in a pseudo-random function. The freezing allows us to have the same results after each run of evaluation, tuning, or classification using the same dataset, which comes in handy when trying to document the performance that can be achieved from a certain text classifier.

The gathered dataset is then divided into 80% of training records, and 20% of test data. Training records are used under the initial evaluation of a non-tuned text classifier with its default key parameters, and in the tuning process to find the optimal key parameters of the text classifier.

In both processes of initial evaluation and tuning, 80% of training data records are divided into 5 folds, where 20% of the training records are placed in each fold. And by performing a k-fold cross-validation test, 5 separate text classifiers are trained using 4 training folds and a new unique fold of testing records for each classifier.

Figure 10 below shows the recommended split plan of training data from the developer community on "scikit-learn.org" **[28]**.

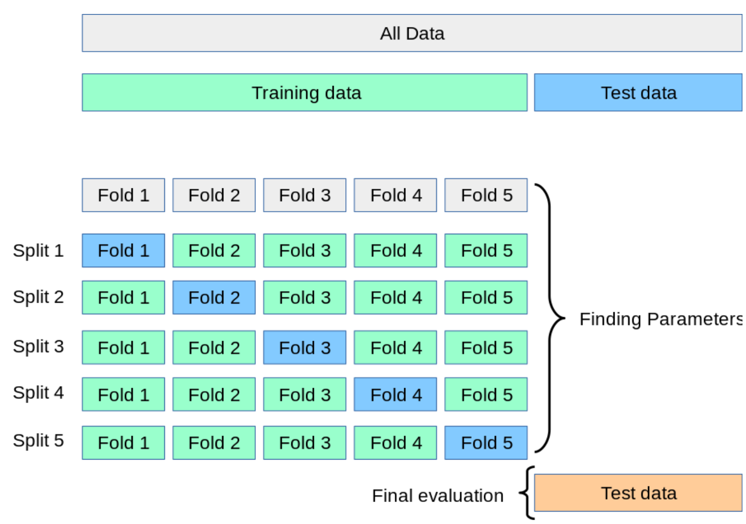

Figure 10, The recommended split plan of a grid search using k-fold cross-validation, obtained from the developer community on ***«scikit-learn.org» [28]****. Training data (Green), Test data (blue), final evaluation data (orange).*

The demonstrated testing process in this section is based on a dataset crawled on the 30th of July 2021, at 12 PM., according to the plan in table 1. However, in the technical solution of this research, a new dataset for training and testing is crawled periodically for each batch of data to help perform the task of influence field detection in the process of building the influence graph.

- **Step – 2: Initial evaluation of the text classifier**

We start by testing the initial performance capability that can be achieved from a non-tuned text classifier using a training dataset from Reddit.

To have confidence in the results of this evaluation, we must make sure that each record in the dataset will participate no more of less than one time in a testing phase. Therefore, it is important to follow the previously explained split plan at the end of step 1 and in figure 10. Then we take the average of the accuracy parameters from each one of the 5 tested text classifiers to have an idea of the initial performance of text classification without any tuning.

After splitting the training records and training the 5 text classifiers, their average performance results are presented in Table 2.

Table 2, Initial performance evaluation of the text classifier.

Metric	Average of measured values
Accuracy	0.692
Precision	0.698
Recall	0.692
F1-Score	0.691

- **Step – 3: Tuning the text classifier**

When examining the results of the previous initial evaluation of the text classifier, we notice very close values of accuracy, precision, recall, and f1-score parameters at nearly 70%. This can be improved by tuning the key parameters of the pipeline algorithms used in this text classifier. The most important of these key parameters are stated in table 3 along with their significance to the text classifier.

Table 3, An overview of the tested key parameters used for tuning the text classification model.

Model Parameter	Tested Values	Explanation
vect__ngram_range	(1, 1), (1, 2), (1, 3), (1, 4), or (1, 5)	Whether to use words of unigrams, bigrams, trigrams, 4-gram-sequence or 5-gram sequence in vectorization
tfidf__use_idf	True or False	Whether to use Term Frequency-Inverse Document Frequency or not
clf__alpha	(1e-1), (1e-2), (1e-3), (1e-4), (1e-5), (1e-6), (1e-7), (1e-8), (1e-9), or (1e-10)	A penalty parameter is used in the SGD[14] classifier **[29]**.

To find the optimal parameters for the text classifier, we run an automated test of cross-validation. In this test, we try every possible combination of the tested parameter values in table 3. Also, at each trial, we partition the dataset according to the split plan visualized in figure 10.

The process of cross-validation must perform 100[15] trials, one for each possible combination of parameter values in table 3. And in addition, 5 text classifiers will be built at each trial according to the partitioning plan in figure 10. This gives a total of 500[16] evaluated text classifiers.

We then choose the key parameters of the text classifier that has the highest f1-score. These parameters are most likely to optimize the performance of the text classifier in future production use. The optimal values of this tuning process are viewed in table 4 below.

[14] Stochastic Gradient Descent **[29]**.
[15] 100 = (5 test n-gram parameters) X (2 tested use tf-idf parameters) X (10 tested SGD penalty parameters)
[16] 500 = (100 tested parameter combinations) X (5 splits of training records in 5-folds)

Table 4, Achieved optimal values of key parameters after tuning the text classifier.

Metric	Measure Value
Best f1-score	0.706
vect__ngram_range	0.001
tfidf__use_idf	True
clf__alpha	(1, 1)

Using these optimal parameters, we manage to increase the f1-score to 0.706 from 0.691. Although this might be a low improvement among the community of artificial intelligence, it is still a good score considering many factors and priorities such as the need of having a dataset that is dynamic and easy to update for use in the training of a text classifier.

Another important question to ask here is, what is a good score under testing and tuning in artificial intelligence? This depends on what we are trying to achieve from this text classifier.

We are only seeking the use of this text classifier to estimate the field of influence between users based on the text in their activities. A score of around 70% is satisfying enough to proceed into using this text classifier for building the influence graph and perform an evaluation on its ability to distinguish between different categories of topics in a crawled dataset.

A deeper picture is drawn in the upcoming evaluation section where an evaluation will be carried out on the influence graph model to reveal its ability to detect and distinguish as many target categories as possible. But first, a final evaluation is performed using the tuned text classifier with the 20% unseen test data showed on the right lower corner of figure 10 in orange color.

- **Step – 4: Final evaluation of the tuned text classifier.**

After tuning the text classifier, we test it using the 80% training data and 20% unseen test data according to figure 10. And with using the optimal parameters obtained from the tuning process in Step 2. Then, we keep track of the actual and predicated category of every testing record which is used to produce an informative classification report shown in table 5.

The classification report indicates a weighted f1-score average of 0.746 with the highest f1-score for the sport category at 0.829 and the lowest score to the entertainment category at 0.706.

The lowest score category might have been affected by the big interference of multiple aspects of other categories into entertainment, which can cause the model to falsely classify records that should belong to the entertainment category and not to other overlapping categories and vice versa.

Table 5, The classification report after testing the tuned text classifier.

Category	Precision	Recall	F1-score	Support
Sport	0.859	0.802	0.829	76
Economy	0.753	0.68	0.715	72
Politic	0.718	0.708	0.713	72
Technology	0.725	0.804	0.763	82
Entertainment	0.688	0.726	0.706	73
Weighted Average	0.749	0.746	0.746	375
Macro Average	0.748	0.744	0.745	375

Examining the confusion matrix in Table 6 below, we observe that around 53% of falsely classified test records stands between the 2 categories of technology or entertainment on the one side, against politic or economy on the other side. And around 21% of incorrect classifications are between the more distinct categories of sport, economy, and politic.

This confirms that both the technology and entertainment records are often falsely classified to belong to the category of either politic or economy and vice versa, which tells us that this text classifier performs better when classifying generally distinct categories with little in common rather than classifying categories that might overlap.

However, and because of having an average weighted and macro f1-scores of 0.746 and 0.745 respectively, we can be confident in the performance of this text classifier and the quality of the dynamic training datasets. Which makes this technique suitable for use in building a social influence model, with good potential of detecting the right distinct category of influence.

Table 6, The confusion matrix after testing the tuned text classifier. This matrix shows correctly classified test records (Green), falsely classified test records (Red), falsely classified test records due to the overlapping between technology or entertainment on the one side, against economy or politic on the other side (Dark Red).

Predicted (below) / Actual (Right)	Sport	Economy	Politic	Technology	Entertainment
Sport	61	2	2	4	7
Economy	2	49	5	8	8
Politic	4	5	51	5	7
Technology	0	6	8	66	2
Entertainment	4	3	5	8	53

In the next section, we will evaluate the performance of this text classifier under the process of forming the influence graph using another batch of real-life data crawled from Reddit. By this, we aim to investigate how well this classifier is contributing to the objective of identifying different influence edges in the influence graph.

V - G. Evaluation of Influence Field Classification

Now that we have investigated the performance of our text classification model and tested its capabilities in classifying different text submissions from online social media into multiple topics, we take in use this text classifier into the previously detailed algorithm of determining the field of influence between two persons based on the combined texts of the main submission, the parent activity, and the child activity.

In this evaluation, we examine the different predicted fields of influence between the participating network users by focusing on the influence edges regardless of which users are connected to those edges. This is because the influence fields are stored in the influence edges and not directly in the nodes representing the authors in the dataset.

The setup of this evaluation starts by crawling the top 3 newest submissions on 5 selected test subreddits other than these in the crawling plan of training records specified in table 1. Each one of these subreddits is strongly associated with one target category in the training dataset of our text classification model. Then we proceed by building the influence graph and record how many influence edges are learned from each subreddit, and compare this result with an observation of the shares between predicted categories of edges in the influence graph.

A data crawl is performed to extract information about the 3 newest submissions on the 5 test subreddits shown in table 7 below, noticing that the test subreddits are not used in the training dataset of the text classifier which was previously specified in table 1. The crawling of the data batch used under this evaluation was performed on the 31st of July 2021, 12 PM.

Table 7, shows the evaluation plan to build an influence graph from submissions of test subreddits, and a crawled training dataset according to the plan in table 1.

Topic	Test Subreddits	Training Subreddits
Economy	Finance	Economics
		economy
		business
Technology	research	technology
		science
		Futurology
Sport	NBA	sports
		olympics
		worldcup
Politic	worldnews	politics
		PoliticsPeopleTwitter
		elections
Entertainment	Cinema	movies
		comedy
		culture

Now that we have defined our evaluation plan to test the performance of influence field detection in the influence graph, a text classifier is tuned and evaluated as described in the previous section. This text classifier was able to achieve an optimized f1-score of 0.733 using this batch of crawled training data. The text classifier was integrated to be used in the process of building the influence graph as described earlier, and the testing results are visualized in figure 11 below.

Here, we observe that we have targeted an equal amount of 3 submissions on each subreddit, where each of the 5 subreddits having a share of 20% of the total number of crawled submissions.

However, the number of comments on each subreddits might differ, which we see clearly in the middle plot of how many influence edges were detected from each subreddit. An example here is the Finance subreddit that is having the highest influence activity of 79.5 %.

Finally, we examine the shares of predicted topics in comparison to the shares of detected influence from each subreddit. We notice a high prediction share in the category of economy at 67.3%, which is expected since the Finance subreddit is providing 79.5% of the total edges in the influence graph. This same observation is also the case for the sport category and the NBA subreddit.

But in the case of the 3 other categories of technology, politic, and entertainment, they have slightly bigger shares in predictions compared to their shares in influence detection. This might have multiple causes such as overlapping, or detection of derailment in discussions under the comment thread, which can be dealt with as an improvement of the text classification ML model.

By performing this evaluation, the prediction model of this research was applied to a realistic scenario, and we have observed a series of highly matching results between what we expect from influence fields and what the text classifier has helped to reveal during the process of building the influence graph.

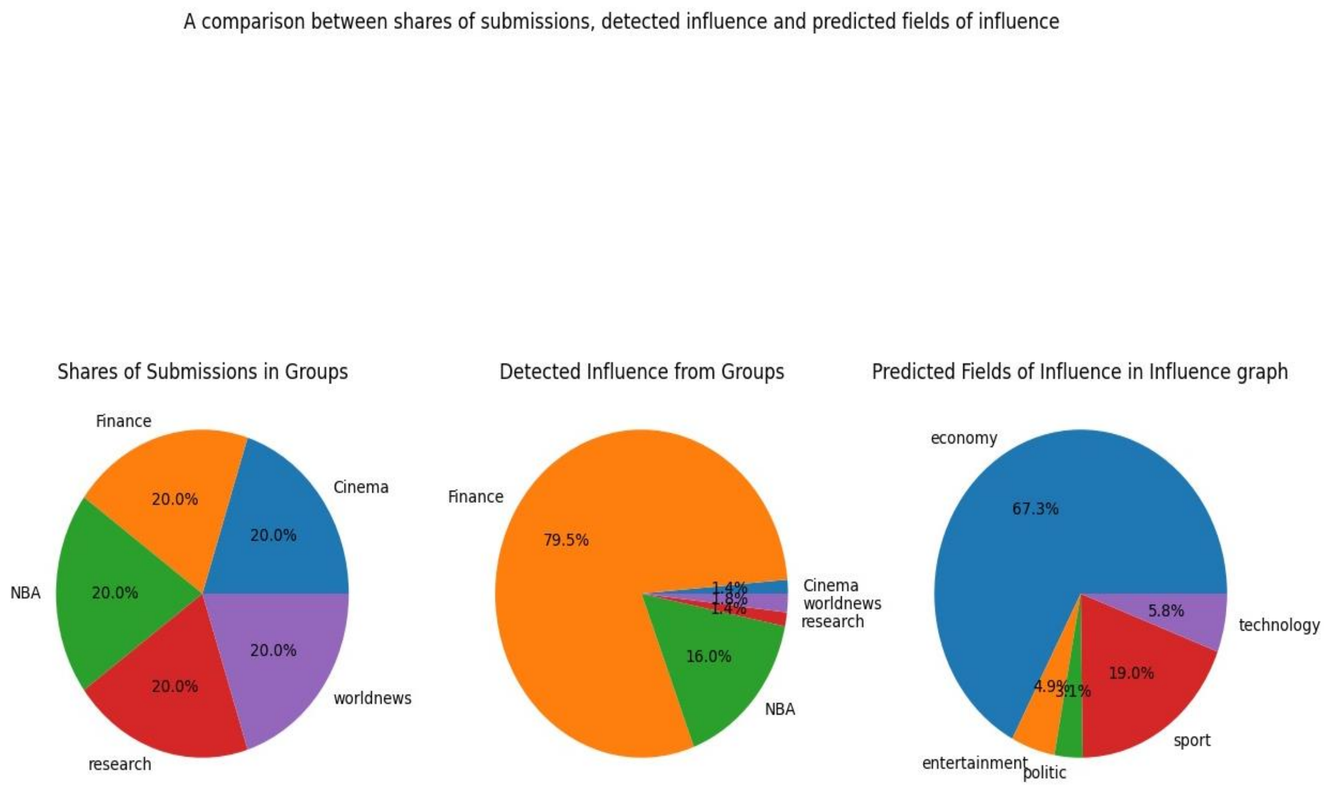

Figure 11, The left pie plot shows the shares between targeted submissions of each evaluated subreddit, the middle pie plot shows the percentage of learned influence edges from each evaluated subreddit, and the right pie plot shows shares of predicted topics recorded in the edges of the influence graph.

V - H. Introducing Centrality Measures

In search of identifying the top influential persons in a social network graph, many techniques can be used to distinguish each user's ability to influence others by assigning each person a certain rank based on the user's connections to other people in the influence graph.

Graph centrality is a well-known technique referring to a group of algorithms that gives each node a calculated rank of importance relative to other nodes in the same graph.

Every centrality algorithm differs from other algorithms in how to calculate the ranking of nodes based on the objective of the algorithm whether it favors the direct links between neighboring nodes or goes beyond the neighborhood to examine the flow of possible paths throughout the entire graph.

In the case of this research, we have an influence graph, representing users as nodes and influence between them as directed edges. Here, centrality measures have the potential to identify the power of each person in this influence graph, both in terms of the user's ability to influence others and his/her contribution in transferring influence from one user to another.

There exist a wide variety of available algorithms used to calculate centrality measures, some of these focus on the direct influence connections to other user nodes while others examine the different paths between nodes that might not have a direct influence connection.

In this project we are going to implement 3 different centrality measures, starting with the connection-based outdegree centrality that focuses on the outgoing edges from a certain node.

Following with an implementation of the betweenness centrality which looks for the occurrence of each user node in the possible shortest paths between any two nodes in the graph.

Then, we will compare the results from each of the previous centrality measures to the Authority and Hub centrality measures of a link-based algorithm called Hyperlink-induced Topic Search (also popular under the name of HITS), this HITS algorithm is considered to be more advanced and complicated than the simpler algorithm of degree and centrality measures.

- **Outgoing Degree Centrality**

In a directed graph, degree centrality is based on the direction of connected edges to each node. It can be divided into 2 segments of calculations; the first counts the number of outgoing edges from a node, and the second counts the number of ingoing edges into the same node.

Although both indegree and outdegree counts are often summed up to output the node's rank, in our directed influence graph, influence edges between persons indicate the influence of the source person on the target person, and because we wish to rank the top influential persons in the graph, the outdegree centrality is a cleaner measure of revealing the power of influence from each person node on its neighbors, regardless of how many other persons does influence a certain influencer.

The introduction of degree centrality to the influence graph model of this research is inspired from a hands-on approach into the use of centrality measures in the analysis of social media **[12]**, which can be accessed to gain more details about how to apply the measure of degree centrality to the networks of social media.

Figure 12 at the end of this section shows an example of an influence graph, where degree centrality measures are calculated for each person in the graph.

We notice that the top influential persons according to the outdegree centrality are "User B" and "User C" since both have 2 outgoing influence edges. While all other users except "User E" have a rank of 1 indicating the only outgoing influence edge they have. "User E" on the other hand is only a receiver of influence and does not influence any other persons which give him the lowest influence score of 0.

- **Betweenness Centrality**

Betweenness centrality is more concerned with the participating role of each person node in all possible shortest paths between any given two persons in the influence graph. Following this centrality measure allow us to rank each person based on his/her ability to transfer influence from one person to another.

The operation of the betweenness algorithm starts by listing all possible shortest paths in the given influence graph, then counts the occurrence of each person in the connecting nodes between the source and target persons in these paths, not to include the nodes of the source and target persons in this occurrence counting. Each node of a person will then have a scored rank equal to the number of times it occurs as a connecting node in any shortest path of the influence graph.

More details on how to apply the betweenness centrality measure to social media networks can also be viewed from the hands-on approach into the use of centrality measures in the analysis of social media **[12]**.

The calculation of influence ranking using the betweenness centrality measure is shown in figure 12 at the end of this section.

"User F" is elected as the top influential person because this person is located on the shortest paths from both "User C" and "User G" to "User B" and "User E".

In the second place, "User B" and "User C" get a score of 3. While the rest of the 4 non-central users get the lowest influence rank of 0 because they do not have a significant role in transforming influence between other persons in this influence graph.

- **HITS centrality – Authority and Hub with 10 iterations**

The process of the HITS[17] algorithm is divided into two symmetric but different calculations that assign two scores to each node in the graph.

The first calculation is based on the ingoing edges of nodes and gives each node a score known as the authority score. While the second calculation is based on the outgoing edges of nodes and gives each node a score called the hub score.

The hub score indicates the ability of one node to point to other nodes in the graph, while the authority score indicates how much a node is pointed to from different hubs.

These two scores are calculated using what is known as the Authority update rule and the Hub update rule, in both rules every node is initially given an old score of one.

To calculate the new authority scores each node gets the sum values of old authority scores given to the source nodes of ingoing edges. Then, to calculate the hub score, each node is given the sum values of old hub scores given to the target nodes of the outgoing edges, before moving on to the next iteration.

For the purpose of normalization and after each iteration, the given new values of authority scores are summed, and each new authority score is divided by the sum of all authority scores to output a normalized score for each node. The same process is repeated on the new hub scores but by summing up all values of the new hub scores. In the next iteration, the normalized new authority and hub scores will be marked as old scores, and the same two processes of authority and hub score calculations are repeated.

In theory, the HITS algorithm can carry on for an infinite number of iterations, but in practice, both values of authority and hub scores would converge each to their approximate value, then the calculations can be stopped as there will be no need for any more iterations. We are here using a fixed number of iterations set to 10 for simplicity reasons but in future improvements, the number of iterations might dynamically change depending on the given graph by detecting convergence of score values. And to avoid optimization issues an upper number of iterations can override and stop any more iterations.

For more details on how to perform the authority and hub calculations in the HITS algorithm, please refer to pages 7-12 in the journal "Authoritative Sources in a Hyperlinked Environment" by J. M. Kleinberg **[13]**.

The HITS algorithm was originally developed to be used in rating journals and web pages, which can point to each other by using references in their text content. The nature of the influence graph of this research is very similar to the usual application of the HITS algorithm, as both of the nodes and links of our developed influence graph are homogeneous, i.e., nodes do only represent different persons, while directed links between them only represent the social influence between them and nothing else.

Establishing some nodes in the graph as important hubs is an advantage of using the HITS algorithm over the use of the PageRank algorithm, this is because the PageRank algorithm favors older, more established nodes even if more recent nodes are very important. More about the capabilities of the HITS algorithm is investigated in the book titled "Discrete Calculus: Applied Analysis on Graphs for Computational Science" by L. J. Grady and J. R. Polimeni **[14]**.

Based on the similarity between ranking journals and webpages in comparison with identifying important persons in an influence graph, we can use the HITS algorithm to find the top influential persons, along with those persons with a high role of

[17] Hyperlink-induced Topic Search.

transforming influence across the graph from one person to another.

Both graphs at the bottom of figure 12 below show the results of running the HITS algorithm with a limit of 10 iterations to assign each node a hub and authority score.

In the graph at the right bottom, both nodes of "User B" and "User C" were given the highest hub score indicating their ability to influence other nodes in the graph by pointing to them.

While examining the authority scores in the left lower graph reveals that "User F" is the most influential person in the graph. This is highly expected as a good authority in the influence graph is a person that is reached from multiple different strong hubs which were found to be "User B" and "User C" in the right lower graph of figure 12 showing the hub centrality scores.

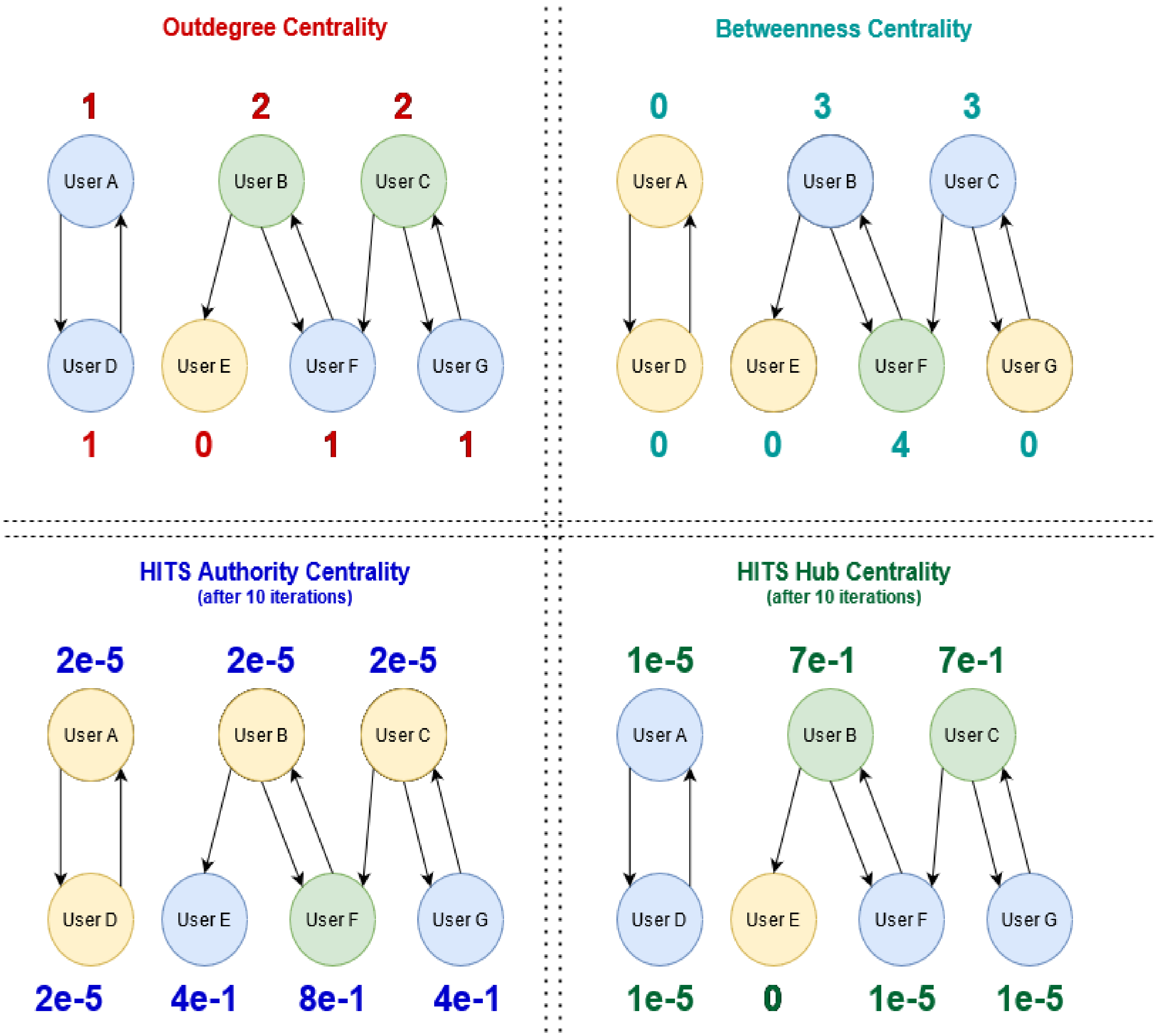

Figure 12, A ranking example using 4 unweighted and directed centrality measures; outdegree centrality (upper left corner), betweenness centrality (upper right corner), HITS Authority (lower-left corner), and HITS Hub (lower right corner). Where top-ranked influencers are shown in green nodes, middle/second-ranked users in blue nodes, and last/third-ranked influencers in yellow nodes.

We hereby conclude that the authority scores help to detect those persons who are more likely to transfer influence between hubs in the influence graph. While the hub scores on the other hand indicate the person's ability to influence others in the influence graph.

Comparing the results of the hub scores to the results gained from the outdegree centrality on the same influence graph, we notice that both measures did yield the same ranking for every person in the graph. This is because the outgoing edges are used to calculate both outdegree and HITS centrality measures. This suggests that the hub centrality is also a direct measure of how influencing a certain person is in his neighborhood in the influence graph. However, the ranking results between outdegree and hub scores might vary when looking at a bigger and more complex graph topology.

Another comparison can be carried out between the betweenness centrality scores and the HITS authority scores on the same influence graph. Here, the same person node is elected as the top influential in each measure. However, the betweenness algorithm tends to favor nodes of persons at the heart of the graph as they often help connect distant persons located at the end of influence paths, while the HITS authority measure tends to favor those distant persons located at the ends of influence paths.

At this point of insight, we notice that each centrality measure has a series of features when used for detecting the most influential persons in a social graph, and no one centrality measure can stand out to give the best picture. This makes it essential for the analyzer of an influence graph to understand the capability of each centrality measure that enables this analyzer to make the best decision on which person is to consider being the most influential in the influence graph. The table below summarizes those previously discovered and discussed features of the various centrality measures.

Table 8, A comparison between 3 centrality algorithms; outdegree, betweenness, HITS – Authority, and Hub. This comparison is based on the observations of ranking performed on the example influence graph in figure 12 and the technical implementation of this research.

	Outdegree	Betweenness	HITS Hub	HITS Auth
Reveals	Direct Influencers	Influence transformers and connectors	Direct Influencers	Influence transformers and connectors
Favors	Number of outgoing influence edges	Persons at the middle of a connected segment of the graph	Number of outgoing influence edges	Persons at the ends of a connected segment of the graph
Simplicity	Easy	Medium	Complex	Complex
Interpretation of results	Easy	Medium	Difficult	Difficult
Running time	Low, because it only requires one iteration over the edges in the graph.	Medium, because graph traversal is needed to find the possible shortest path between every two persons in the graph	High, because it requires multiple iterations to achieve convergence	High, because it requires multiple iterations to achieve convergence

V - I. Centrality Measures vs. Influence Edge Score

Now that we have analyzed the application of some centrality measures on the influence graph, we can use the ranking results on each person in the graph to establish an understanding of which persons are to be considered the most influential among others in the same influence graph.

Before moving forward in analysis, it is important to highlight the similarities and differences between using the previously introduced scores of outgoing influence edges from one person to another and using their centrality ranks as a measure of influence strength.

Centrality measures are calculated on a larger scale than the scores on influence edges because centrality measures either examine a group of connected nodes like the node's neighborhood in the graph, which is the case when using the outdegree centrality, or it can take on the global scope of the graph by examining possible paths in it, as for the betweenness centrality.

Scores of influence edges on the other hand are more personal as they indicate the source and receiver of influence and how strong each influence is, not to forget that to include a certain influence between two persons in the influence graph, at least one interaction between these two persons must be registered.

This leaves us to wonder what to use to be able to locate a good influencer. The best answer is to combine the use of some or all the available measures to establish a decision, without forgetting to evaluate the objective of the analysis itself, i.e., if we want to find the top influential persons that help spread influence throughout the network even though they are not necessarily the source of this influence, then the betweenness and HITS authority measures should have a greater weight in the analysis. And on the other hand, if we would focus on the isolated ability of persons to directly influence others, then the outdegree and HITS hub centrality would help draw a better picture.

At the same time, influence edges can be visualized in any intuitive form like edge thickness or color, which enables analyzers to easily point out the area where the highest scores of social influences occur. And then look at the centrality rankings to make a stronger decision about which persons are to be elected as the most influential.

An example scenario is given in figure 13 below based on crawling the top 3 newest submissions on the top 3 most popular subreddits on the 31st of July 2021, 12 PM.

Influence edges of high total scores (i.e., the sum of interaction, upvotes, and activity scores) are thicker than those with lower total scores, while in the same graph, larger and darker nodes with less color transparency indicate a higher centrality rank.

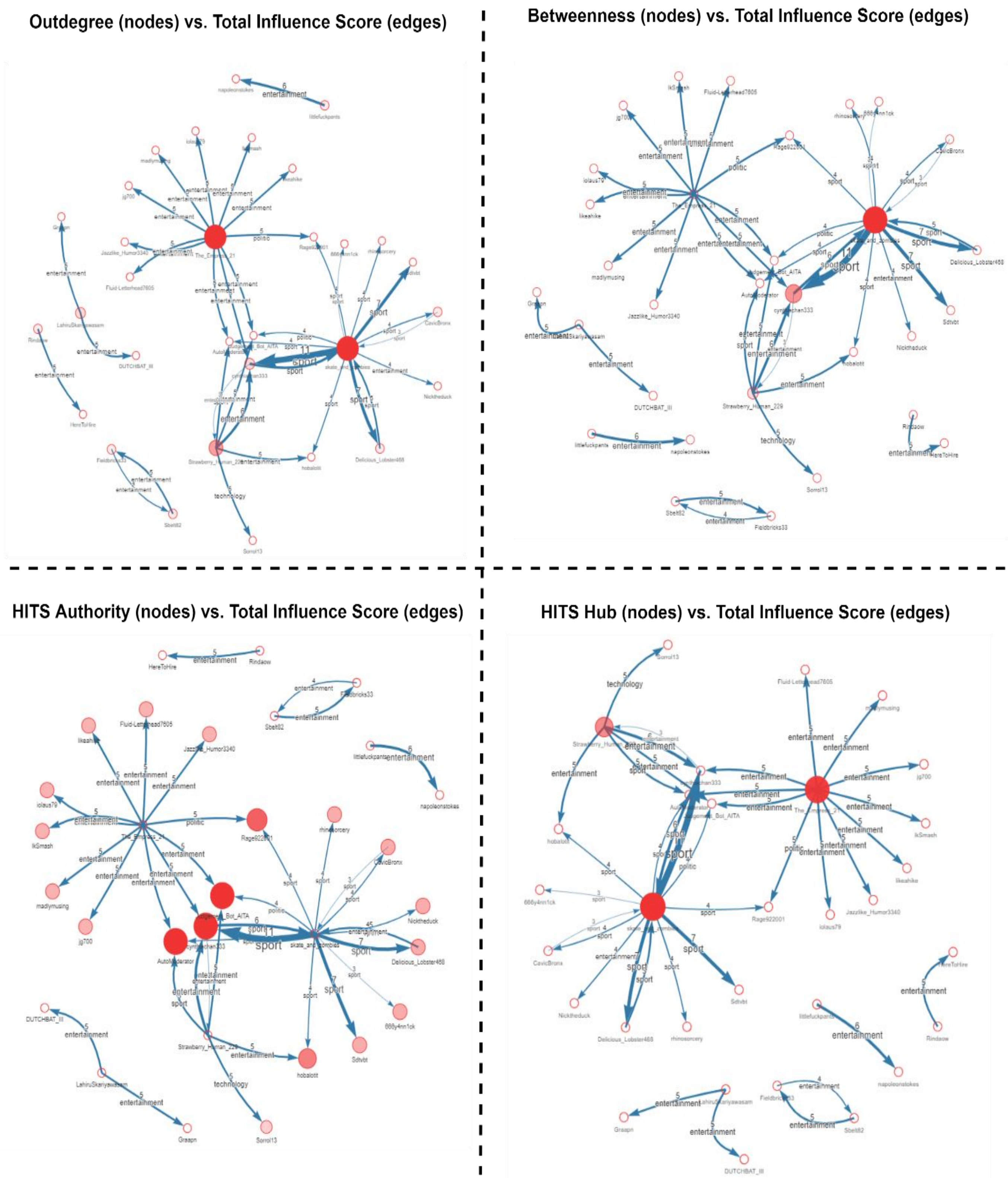

Figure 13, A visualization of an influence graph to compare between the use of node size and transparency to indicate the centrality-based rankings of outdegree, betweenness, and HITS centralities, against the use of edge thickness to indicate the total influence score between people in the same graph. In case these graphs are unclear, a bigger version of each graph is to be found in Appendix – C to F.

Using the degree centrality to rank persons in the upper left graph of figure 13, we notice that one of the top-ranked persons does not have that strong influence compared to the other top-ranked person. This is because we are using the unweighted version of outdegree centrality, as we wish to relax the weight of direct influence and focus on the position of a person's node in the influence graph.

The same results are also reflected on the graph where the HITS hub centrality ranking is performed, but with greater separation between highly ranked persons and others with lower centrality values, which is most likely due to the use of normalization of HITS measures.

On the other side and examining the upper right graph in figure 13, which visualizes the rankings of betweenness centrality, we notice as expected that it tends to give higher centrality scores to those persons who can connect other remote persons in the shortest path. This is however not the case for the graph using the HITS Authority ranking at the lower left side of figure 13, as persons at the ends of an influence path tend to gain a stronger rank, which is good if we are trying to find the most influenced persons in the influence graph.

Table 9 below shows the top 3 rankings of the 4 graphs in figure 13 above, using different centrality measures for each graph.

The 3 measures of the outdegree, betweenness, and HITS-Hub centralities had ranked the same person on top, marked with the green-colored cells. However, they differ with ranking different persons in the top second and third place, which is a satisfying result, because we are expecting centrality measures to have a certain similarity in detecting the most direct influential persons in the graph, but also differ in lower rankings to reveal possible hidden information in the influence graph and help bring it up to the surface.

When comparing the betweenness rankings to the HITS authority rankings on the same graph shown in table 9 below, we notice a single common selection of one person in the first rank using the HITS authority, and in the next second rank using the betweenness centrality, marked with the blue-colored cells.

Other than that, there is a great variety in ranking influencers which is also the desired result to reveal as much information as possible using different measures without having a total difference in their outcome.

Now that we have reviewed a practical example of analysis and investigated the common and uncommon features between edge scores and centrality measures. We shall be precise that to correctly rank the most influential person in the influence graph, it is crucial to understand the features of the used centrality measure and its impact on the final ranking results. Used centrality measures should be studied hand in hand with the individual influence edges and their selected score, and perhaps their predicted field of influence. This might sound to be a complicated process of analysis but when using the right visualization tools, the display of the influence graph can become more intuitive and easier to understand as we have seen earlier in figure 13, and by using the demonstration website at https://smia.uis.no.

Table 9, The ordered rankings of the top 3 most influential persons in the influence graphs of figure 13 using the 4 centrality measures; outdegree, HITS Hub, Betweenness, and HITS Authority. Colored cells are used to mark the same persons in the ranking across these 4 centrality measures.

Rank	Outdegree	Betweenness	HITS Authority	HITS hub
1	skate_and_zombies	skate_and_zombies	Judgement_Bot_AITA	skate_and_zombies
	The_Empress_21		AutoModerator	
			cynthiachan333	
2	Strawberry_Human_229	cynthiachan333	Rage922001	The_Empress_21
3	LahiruSkariyawasam	Strawberry_Human_229	hobalotit	Strawberry_Human_229
	cynthiachan333			

V - J. Testing & Evaluation of The Activity & Influence Graph Models.

An important desire in any graph is to be informative, easy to read, and easy to visualize, we hereby wish to test how informative the activity and influence graphs are in visualizing the picture of influence flow in a studied dataset.

We start by visualizing the activity graph on a dataset crawled from Reddit on the 30th of July 2021, at 12 PM. This dataset targets information about the top 3 newest submissions taken from each of the top 3 most popular subreddits according to internal Reddit statistics by using the python Reddit API wrapper.

Recall that the activity graph has 3 types of nodes: the submissions, top-level comments, and sub-level comments.

In figure 14 below, we view the activity graph of this Reddit dataset and notice a 62.5%[18] higher concentration of top-level comments having the red color in comparison to sub-level comments with the blue color. This tells us that people in this dataset have low to medium engagement in the threads of submissions. A higher engagement and thus more sub-level comments will most likely benefit the process of detecting social influence in the dataset when transforming this activity graph into the final influence graph.

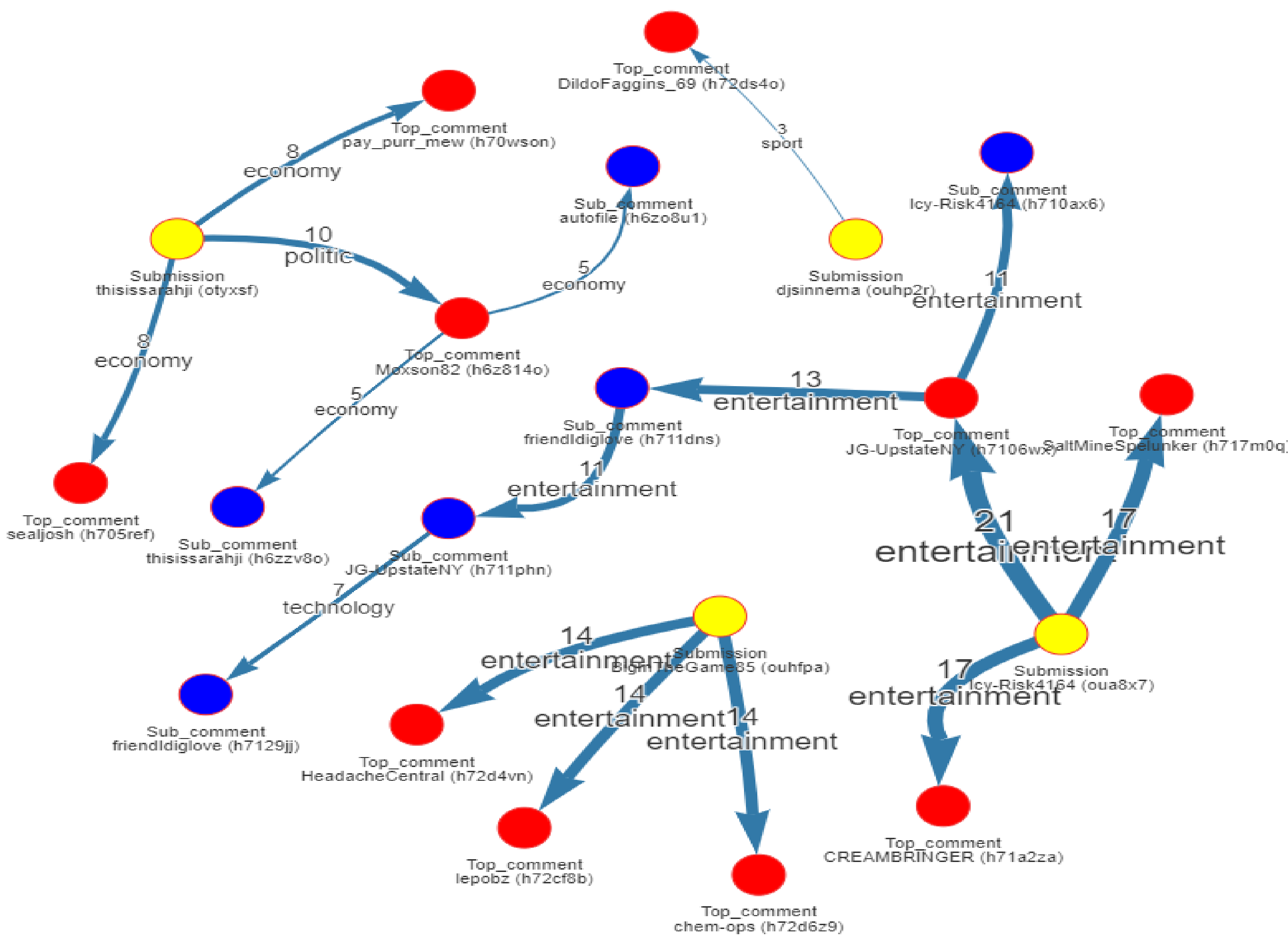

Figure 14, The activity graph shows submissions (yellow nodes), top-level comments (red nodes), and sub-level comments (blue nodes). Along with the score and predicted type of influence on each influence edge.

[18] 62.5 % = ((10 number of top-level comments) / ((10 number of top-level comments) + (6 number of sub-level comments))) * 100

The nodes of the activity graph in Figure 14 represent 20 activities in the crawled dataset from Reddit, having a count of 4 submissions, and the remaining 5 submissions do not contain any comments and therefore are not shown in the activity graph. Also, 10 top-level comments and 6 sub-level comments are founded in this activity graph.

Since the percentage of sub comments is around 30 %[19], it can also be used as an indication of low to medium social engagement from the participating influencers in the dataset. This result is highly visible by taking a quick look at the colored activity graph in figure 14 without having to calculate the exact number of each activity type. This highlights the importance of defining the types of activities to be used under the visualization of the activity graph.

By digesting this previous activity graph and merging its edges to transform it into an influence graph, we obtain the graph shown in figure 15 below. The visualization of this graph takes advantage of the total combined score values to accordingly adjust the thickness of each influence edge, indicating its high or low score value in an intuitive display. This allows analyzers to identify and focus on the detected influence with high scores.

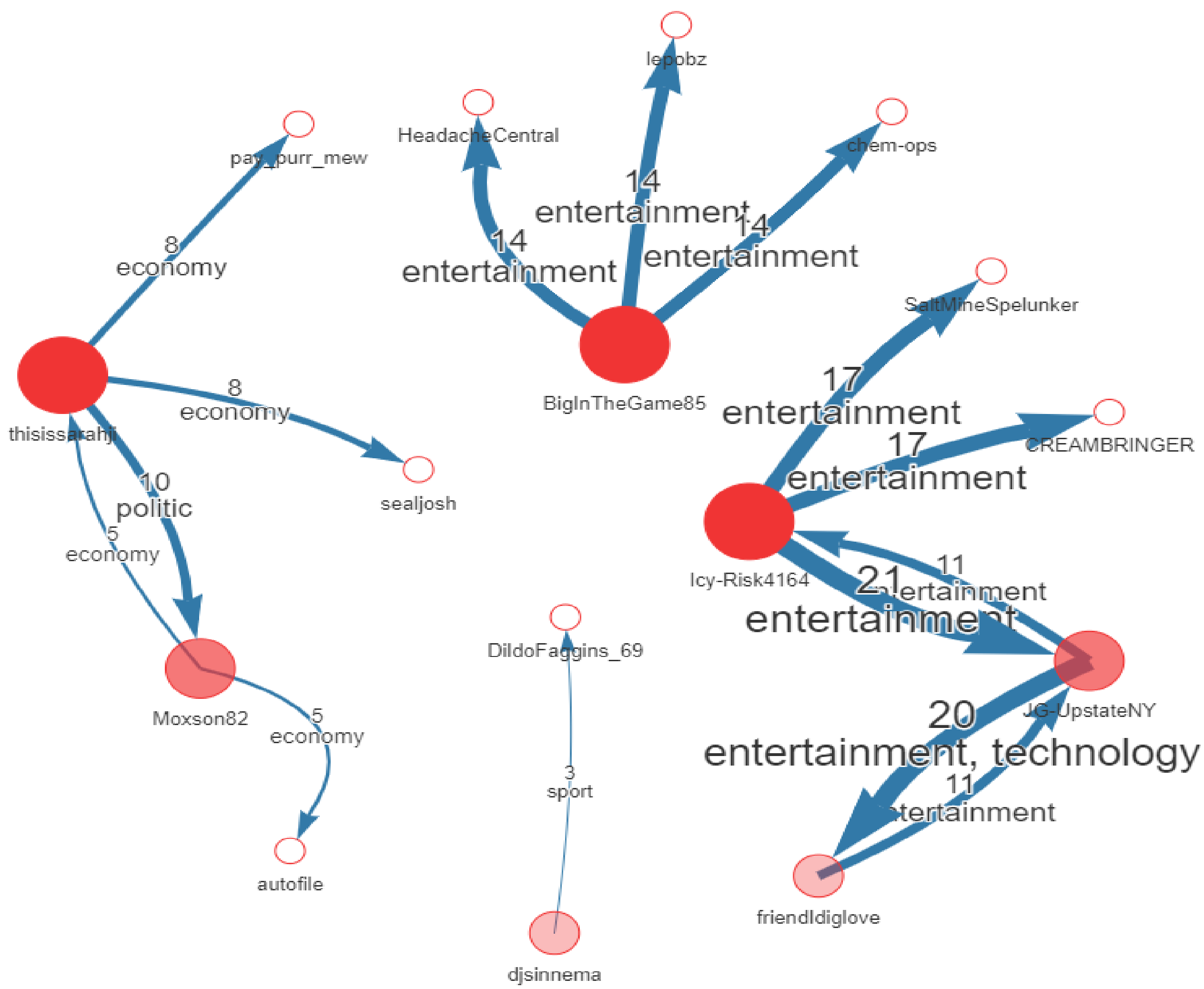

Figure 15, The influence graph shows persons in the dataset in unique nodes and influences between them as edges. A bigger size and color transparency of nodes indicate a higher outdegree rank, and a higher thickness of influence edges indicates a greater total score (i.e., the sum of activity, interaction, and upvotes scores) of influence.

[19] 30 % = (6 sub level comments / 20 activities in total) * 100

In addition to visualizing the strength of each influence, the predicted type of influence is also used as a label on each influence edge, where only one influence type is possible for each influence edge in the activity graph, but due to the merging into the influence graph some edges might accumulate multiple influence types as viewed in the right bottom corner of the influence graph in figure 15, this tells us that this influence is most likely to be in entertainment and/or technology.

Another important feature of the influence graph is the use of unweighted outdegree centrality to identify central influencers and distinguish them from other less influencing persons in the graph, which can be visualized using their calculated centrality value to be reflected on the size and color transparency of the nodes in the influence graph as in figure 15.

The use of unweighted centrality measures allows the analyzer to have a view that is independent of the score of influence edges, as we observe in the influence graph in figure 15, where 3 significantly bigger and darker nodes are observed.

The big node on the left side of the graph does have some significantly lower scores of outgoing influence edges despite its high centrality rank. While the other two big nodes have thick outgoing influence edges and can therefore indicate higher influence from these two influencers on other connected network users.

Looking at the overall collection of different features and tools provided by the final influence graph, it is clear to observe its beneficial capabilities during a process of social influence detection and analysis. Many applications can then benefit from the ease of interpreting the influence graph to determine the strength of influence, its predicted field of influence, and the centrality ranking of network users in the influence graph.

V - K. Future Improvements of The Influence Graph Model

Online social media is developing rapidly, and therefore, it is a necessity to keep the previously developed model of influence graph up to date with the newly added functionalities and features on social media platforms. In the future, there should be a continuous improvement process that evaluates what online user functionality to be used for detecting social influence, and thus guaranteeing the capability and relevance of the influence model in the future analysis.

One example of a possible online user functionality to be implemented in the future is the reaction trend on submissions and comments, this feature has gained a great deal of popularity among users of online social media, and in many cases, it outruns the effect of writing comments in influencing other users due to its easiness and agility.

Improving the influence model is not only limited to adding support for more online user functionalities, as it is also possible to further develop the modeling algorithm that produces the influence graph to implement a more advanced mathematical formula to calculate the weight of influence, rather than using the basic addition when combining scores with different types such as the total score, which is now a simple addition of upvotes, interaction count, and the number of descending activities in the branch of the parent activity node.

In addition, more scoring techniques can be developed and used in the future, an example here is to modify the activity score to only count the direct children of each activity and use this count as a weight for the detected influence, which can have a better reflection of the influence impact on the same level in the activity thread tree.

In the aspect of classifying the types of influence, we can modify the category set we are using in this project by adding, removing, or changing any categories, this is easily done in the technical solution of this research and only requires a simple edit of the configuration array for which subreddits to crawl and what topic to be associated with.

The dynamic text classifier can also be replaced by a more advanced classifier that can predict the topic of influence based on submitted pictures in addition to text.

Another way of improving this classifier is providing a specialized high-quality dataset for topic classification or testing the performance of alternative ML algorithms in sentimental analysis.

Furthermore, it is also possible to consider taking into account the entire discussion on a submission including all comments on it and then give every influence detected from the thread of this submission and its comments a common estimated field of influence.

Drilling down into our already implemented text classifier, we can also look for more improvement for accuracy and F1-score upon tuning and evaluation by adjusting the tuning range of key parameters or change some of its methods, such as using the naïve Bayer classifier instead of the currently used stochastic gradient descent classifier. However, it is important to compare the results on the very same training dataset, as some differences in the training datasets can lead us to falsely choosing the least effective text classifier.

For ranking network users in the influence graph to find their influencing impact on others, it is also possible to integrate more centrality algorithms in a future improvement.

Another way of improving the centrality-based ranking is to take advantage of the weighted influence edges and introduce weighted centrality calculations for a more advanced centrality ranking.

There are numerous ways in which we improve how we detect and score influence between network users in the modeling of an influence graph. The most important rule to keep in mind is the objective of improving the capability of the influence model while keeping the application and its algorithms as simple as possible. Continuous tuning, testing, and change management should help to achieve this objective and keeping the produced model of the influence graph both relevant and informative.

VI. The Technical Implementation

VI - A. Design Architecture

The desired outcome of this project is a technical information system that can identify and classify influencers and their fields of influence periodically, then visualize this information through graphs. This information is to be stored in a database so it can be looked up later and help trace the rise, fall, and evolution of influencers on online social media.

For this purpose, a technical solution was developed hand in hand with the previous research to provide a valuable tool for analysis, which is publicly available using an easily accessible and intuitive web interface. This technical solution consists of four segments corresponding to the nature of the main process taking place in each segment of the system.

The first segment deals with crawling data from a remote social media platform, then reconstructing this data and extracting the values of the required attributes according to the previously established Entity-Relationship model at the beginning of this research in figure 2. This data is then stored in an archive database where it can be used right away or looked up later in the future to serve as an archive or backup database.

The crawling phase is followed by the process of using this crawled data to build the previously introduced activity graph, then process the edges between its activity nodes to build the influence graph between influencers and store each graph in a separate dedicated graph database; one for storing activity graphs, and the other for storing influence graphs.

Another important segment of this technical solution is the integrated module dedicated to monitoring crawling activities and generating informative plots about each influence graph, like the distribution of different scores in influence edges and the percentage share between different predicted fields of influence. The generated plots are to be stored on the file system of the application host and can be retrieved using the web interface of this application.

The last segment sets up a front-end web interface for this application by spinning up a dedicated web server, where constructed activity and influence graphs can be retrieved and filtered from their respective neo4j databases, along with the option of retrieving statistics about each graph and the initial crawling process.

Figure 16 sketches the system design architecture of this technical solution, this architecture is oriented to the expected data flow from a social media platform to the final analysis tools provided by the web interface.

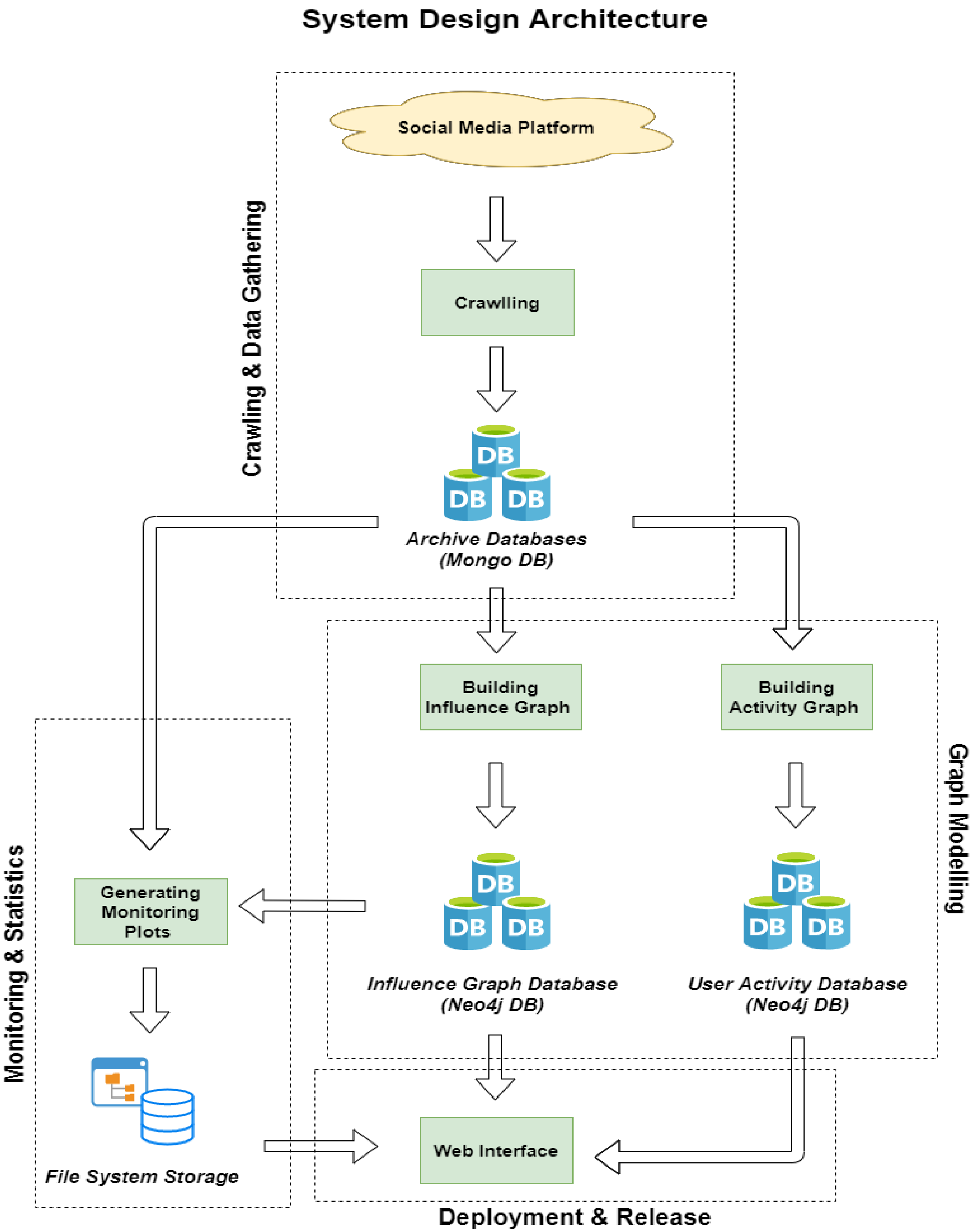

Figure 16, The system design architecture of the technical solution of this research.

To store crawled data in a more general and modern data structure, we take into use the NoSQL mongo document database, which stores information about the groups, submissions, and comments in a simple data structure that is very similar to the structure of JSON[20]. The JSON structure has proven capabilities in modern IT systems due to its ease in both readability and programmability.

After storing the crawled data on the mongo archive database instance, this same data can now be retrieved and used internally to build the activity and influence graphs, then each one of these graphs would be stored in a separate graph database.

The technology of neo4j graph databases is used in this technical solution, which gives us an SQL[21]-like query language that uses out-of-the-box highly optimized graph algorithms to perform a series of important tasks for our application such as calculating the centrality rankings of influencers upon constructing the influence graph, and also finding and retrieving certain segments of the influence graph based on parameters specified by the users of the web interface, examples of such features are filtering the influence graph by a score range or certain types of influence.

A demonstration site of this solution is published under the domain https://smia.uis.no and is expected to be alive until the 15th of June 2022. This site performs two separate Reddit crawls each day at 12 PM. Oslo time; one that crawls the top 3 newest submissions on the top 3 most popular subreddits, and the other crawls the top 3 newest submissions posted on 5 targeted subreddits highly associated with different fields of influence.

The source code of this technical implementation is publicly available on GitHub.com through this link https://github.com/MohammedGuniem/social-media-influence_analyzer, with a proper README.md file for details on how to set up a development or test, and production environments of this technical solution.

Various technologies were used in this technical system, these technologies are documented in [58]-[84].

VI - B. Crawling & Data Gathering

The phase in the technical solution where data is downloaded from remote social media and stored in the local archive databases is called the crawling phase.

This phase is the endpoint at the back of this application, and it is divided into two main processes; the first process is about fetching data from the remote endpoint of a social media platform, then transforming this data to a data structure that satisfies the ground entity-relationship model of this research shown in figure 2 earlier.

The second process of the crawling phase is storing fetched data in the Mongo archive databases using an appropriate storing schema that maximizes the efficiency of reading and writing data, which makes it easier for a database administrator to manage and manipulate data directly on the database server.

Every crawling job made from this application is distinguished by its combination of the following 3 parameters:

- **Network name:**

Which often includes the name of the social media platform from where the data has been read, but another name can also be used.

- **Type of crawled submissions:**

Submissions can be labeled with a certain type, such as new, rising, or controversial submissions. This same parameter can be used to separate different datasets from the same social media platform.

- **Date of the crawling day**

As we wish to build a system that crawls and analyzes data on periodic time intervals, we will include the date of the day where the actual downloading of data took place to identify the different daily crawled batches of data in a continuous daily timeline.

Further on, data from each crawling batch is separated into 4 Mongo archive databases where information about groups, submissions, comments, and training data for influence field classification is stored in its respective database.

In each of these databases the date of the crawling day is used as the collection name, this means that no matter how many times a crawling job is scheduled to run within one day, all data from this day will be accumulated in the same collection making it possible to arrange data in a daily timeline. Figure 17 shows the progress plan of a crawling job

[20] JavaScript Object Notation.

[21] Structured Query Language, SQL.

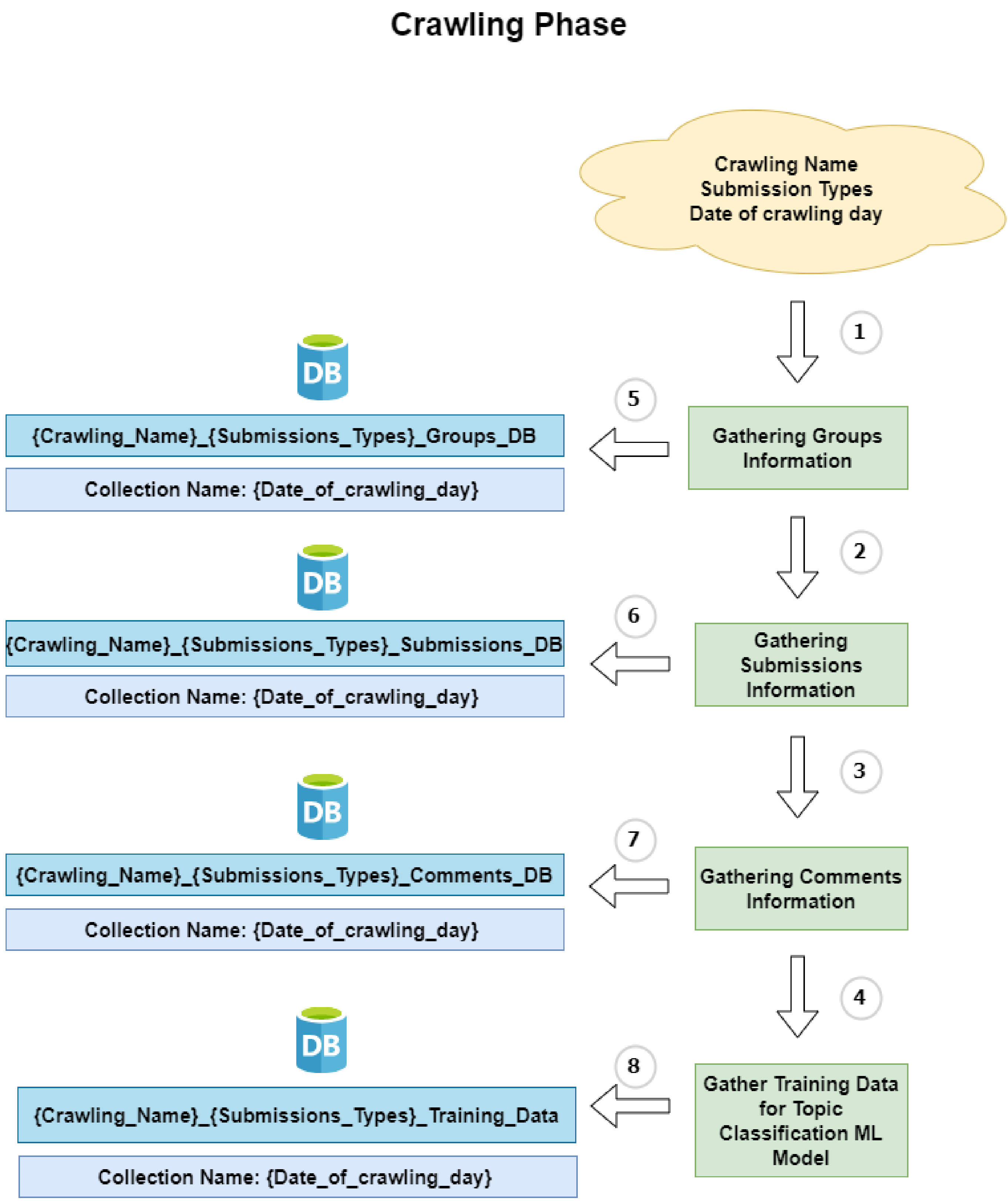

Figure 17, Storage plan of the daily crawled datasets, please note the numbered sequence of crawling and data storage used to avoid any inconsistency in stored datasets.

It is important to notice that all data should be fetched from the remote social media platform before moving on to writing this data in the archive databases. This is to avoid any inconsistency and cascading effects that might result in having an incomplete dataset, because of a failure or loss of connection under the transfer of data from a remote social media endpoint to this application.

To have a mutual coherence between the actual code that performs the processes of the crawling phase and the data stored in the databases, it is recommended to create a crawling class for every targeted social media platform.

Such class should at least implement 4 methods to perform the crawling and processing of data from targeted social media groups, then their submissions and comments along with a targeted training dataset used for influence field classification. Additional methods can also be implemented as an extension if needed.

Crawling classes belong in the “crawling” module located in the “classes” folder of the source code that forms this technical solution.

An example of such a crawling class is the Reddit crawling class “RedditCrawlClass.py” located in the “crawling” module. An instance of this crawling class can be created to be a part of a driver script performing a scheduled driver task that performs crawling, modeling, and other phases in the sequence described by the draw of the system design architecture in figure 16.

The following diagram in figure 18 visualizes the crawling mechanism of new submissions from the top 3 most popular subreddits, which helps in clarifying how the crawling phase can be applied for future crawling targets.

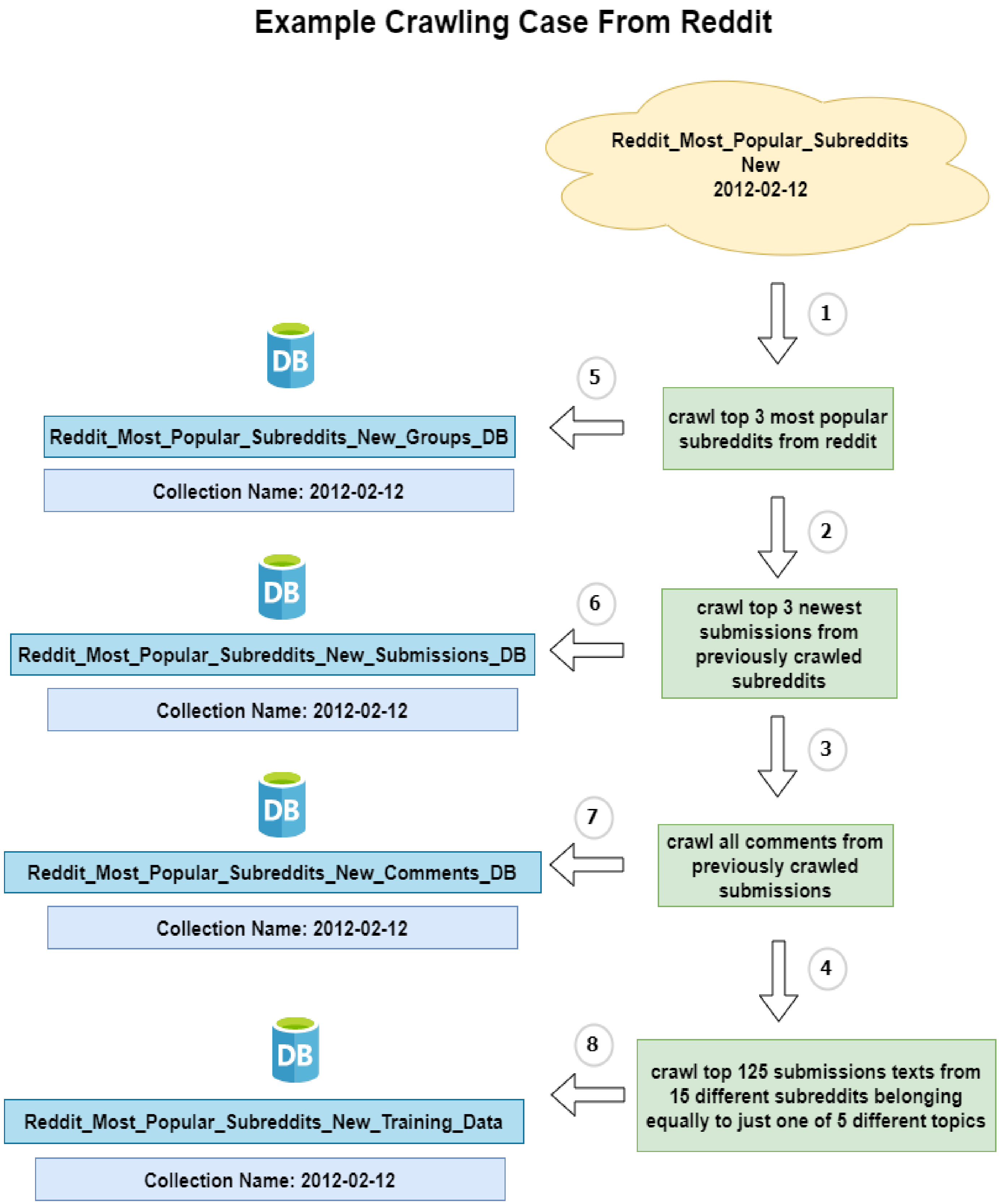

Figure 18, An example of the progress in the crawling phase for crawling the 3 newest submissions on the top 3 most popular subreddits, along with a training dataset for text classification used for the task of influence field detection.

VI - C. Implementation of Graph Modeling

The next phase after processing and archiving the crawled data from a social media platform is to apply our modeling algorithms on this batch of data to produce the activity and influence graphs. Both algorithms are explained with detailed examples under the main section of influence graph modeling earlier in this research.

Although the influence graph is built on top of the activity graph and can be considered as a transition from an activity-oriented graph to a user-oriented graph, it is possible to separate the building of each graph by writing a separate algorithm that directly outputs the respective graph. This is especially important to avoid building the activity graph when there is only a need to build or refresh the influence graph under development, testing or while debugging certain issues.

The next sketch in figure 19 shows the detailed flow of different processes during the graph modeling phase, where in steps 1 to 4, data is fetched from the Mongo archive databases using the unique parameter combination of the crawled batch of data.

Then in steps 5 and 6, an optimized text classifier is built to participate as an influence field classifier when building the activity and influence graphs. Finally, each graph is stored in its respective neo4j database.

The centrality algorithms are set to run on the final graphs after being written to the databases, and then update the centrality properties for each node.

This is a great advantage for using the neo4j graph technology, which has a collection of built-in procedures that can be triggered to activate many algorithms of graph data science, such as calculating the centrality of nodes in a registered graph using a wide variety of centrality measures like the centralities of outdegree, betweenness, and HITS. This helps save time and resources by removing the burden of implementing the different centrality algorithms from the shoulder of the system developer.

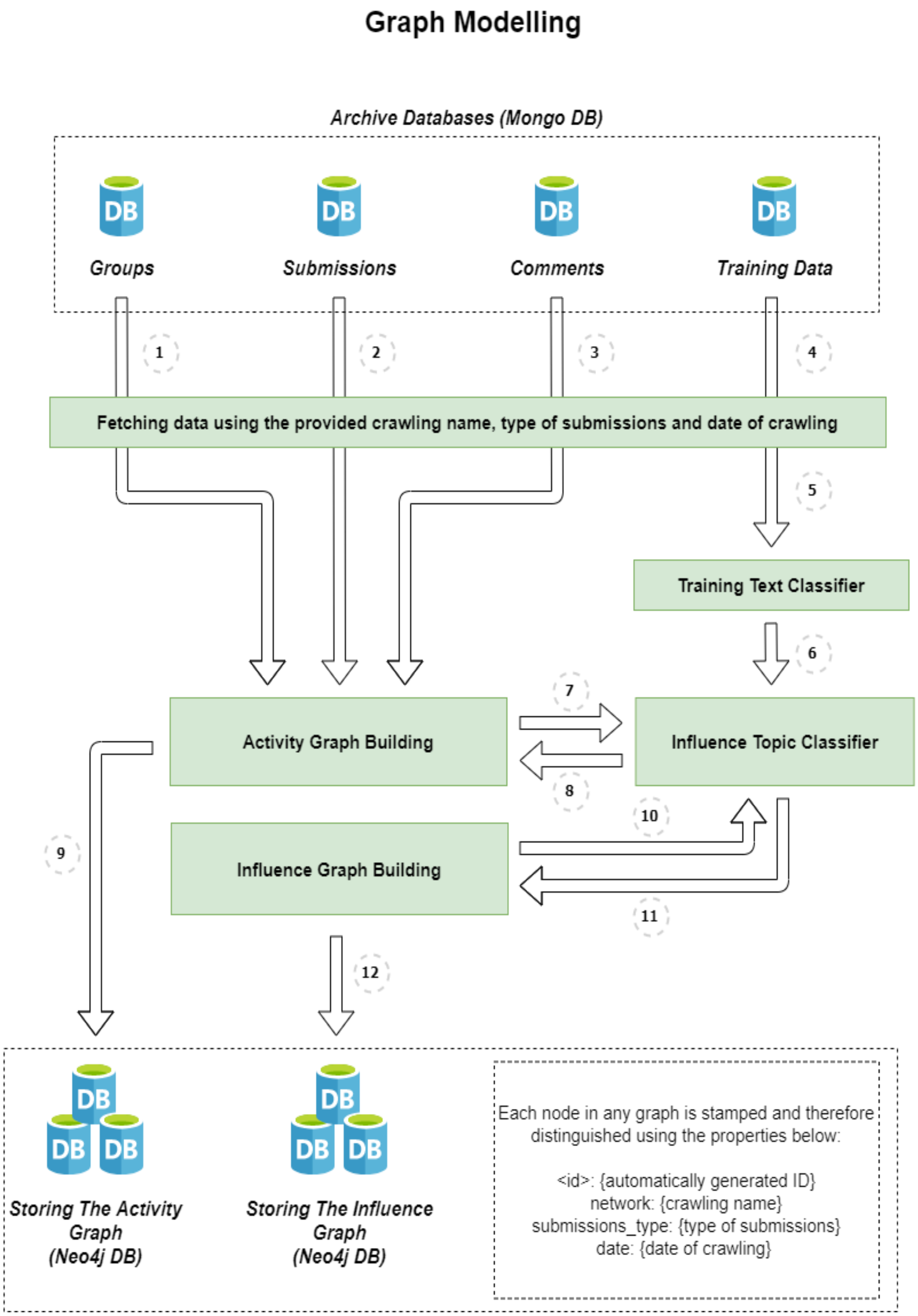

Figure 19, A detailed sequential overview of the different main processes during the periodic graph modeling in this technical implementation.

The source code that implements this graph modeling phase is to be found inside the module called "modeling" in the "classes" folder within the source code. In this module, we find the classes listed in table 10 below with a brief description of their purpose in the technical application.

Table 10, A brief description of the integrated classes in the module specialized for graph modeling within the source code of this technical solution.

Class Script (Class Name)	Purpose
Node.py (Node)	Defines an object that represents the graph node type, with the appropriate attributes of a graph node.
Edge.py (Edge)	Defines an object that represents the graph edge type, with the appropriate attributes and some supplement methods to do important jobs such as updating the score of an edge while building the graph and retrieving the properties of a graph edge.
TextClassification.py (TextClassifier)	This is the object class that provides a text classifier to be used for predicting and classifying the field of influence based on the texts of activities as an input. In this class, evaluating, tuning, and preparing the text classifier are implemented each in its respective method. But before training the text classifier, a tuning process is performed to make sure the produced text classifier is optimized. A reporting method is also included, which returns the classification report and confusion matrix after the final evaluation process of a tuned text classifier. Finally, a classification method is also included in this class to serve the purpose of classifying influence edges during the process of building the activity and influence graph.
GraphModelling.py (Graph)	This is a generic class that holds the common functionalities in building the activity and influence graphs and stores the outputted graph to the respective neo4j database.
ActivityGraphModelling.py (ActivityGraph)	This class inherits from the "Graph" class and implements an algorithm for building the activity graph, it also implements some needed methods especially for this algorithm.
UserGraphModelling.py (UserGraph)	This class inherits from the "Graph" class and implements an algorithm for building the influence graph between activity authors in the provided dataset, it also implements some needed methods especially for this algorithm.

VI - D. Monitoring & Statistical Overview

Most graphs increase in complexity and the amount of information they hold when increasing in size, and the developed influence graph of this research is no exception. It is therefore important to monitor information provided by the influence graph to have a better understanding of its capability in revealing the full picture of influence flow between activity authors.

Monitoring and viewing of continuous statistics about both nodes and edges of daily influence graphs are integrated into this technical implementation and found in the module called "statistics" located in the "classes" folder within the source code.

A class called "Statistics" offers a series of specialized and flexible methods to generate key plots about the amount of time data crawling took, and a general overview over the distribution of edge scores, along with a pie chart overview of social media groups and detected types of influence in the graph. These plots were used earlier in this research as intended in the test and evaluation processes of different aspects in the influence graph.

The generation of statistical plots is triggered in the driver script after the process of modeling. It uses the popular python libraries "matplotlib" and "pandas" to achieve this task, then stores these plots as "png" images on the server file system in a properly organized directory structure. These plots can then be retrieved by analyzers using the web interface of this application.

The web interface also offers other methods for direct information retrieval from the influence graph, such as the centrality reports that show a series of ranking orders of influencers in the influence graph according to the implemented centrality measures.

And not to forget the possibility of viewing the underlaying activity graph, which gives birth to the influence graph and helps explain it by visualizing the distribution of submissions, top-level comments, and sub-level comments in the crawled dataset using different colors of nodes.

VI - E. Deployment & Release To The Web Interface

The deployment and release processes are highly automated for publishing the results of daily built graphs, along with their corresponding statistics and monitoring plots.

The main operations of this process are implemented as a lightweight web server that uses the python "FLASK" framework and offers multiple HTTP(s) endpoints. These endpoints give analyzers access to a powerful graphical tool that includes many useful features such as viewing all influence and activity graphs, and finding an influence path between two persons, or filtering influence edges according to a certain desired range of scores and a desired range of detected types of influence.

Many of these HTTP(s) endpoints can also serve as an API to retrieve information in JSON format by simply adding the "&format=json" key to the same URL. This comes in handy when there is a need to connect or feed data to other systems that might work on the results from this application to increase its value and benefits its current and potential users.

Another feature of the web interface allows the analyzers to view the evaluation and tuning reports of each text classifier used in influence field detection, which is dedicated to a certain daily crawled batch of data on a targeted social media group(s).

This web interface is founded in one script located in the root folder of the source code and called "ui_web_server.py", running this script will spin up a web server after ensuring that the underlaying database services are all up and running and therefore ready to receive queries.

Figure 20 shows a screenshot of the index page of this web interface published under https://smia.uis.no.

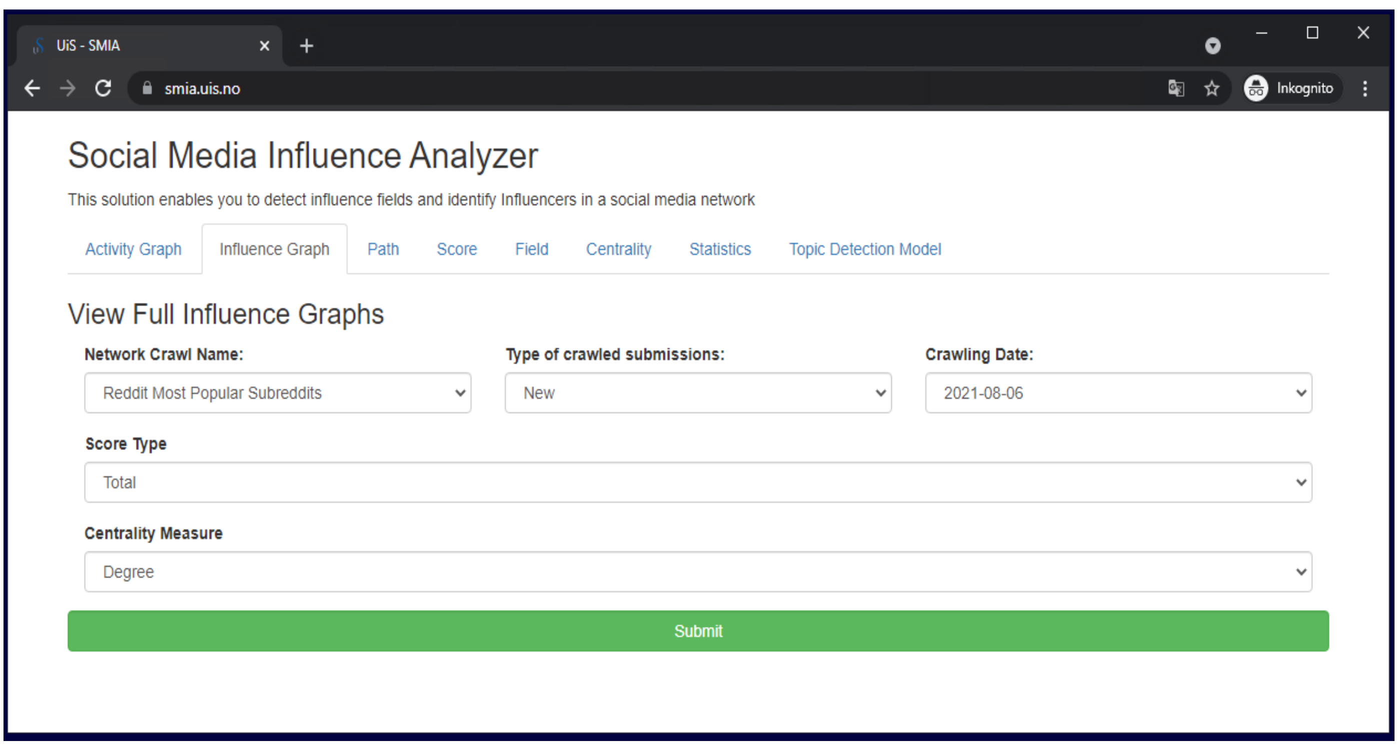

Figure 20, A screenshot of the web interface of this technical solution published under https://smia.uis.no, multiple tools, and features are provided in the navigation tabs of this index page.

VI - F. Building a Driver Script

Now that we have segmented the way this application receives, processes, and retrieves data, it is time to put all these pieces together to produce an example driver code dedicated to a specialized crawling job.

An example of a crawling job is crawling the top 3 most popular subreddits by processing their top 3 newest submissions periodically like for instance once each day.

The technical script of this crawling job is included at the root directory of the source code and is called "reddit_most_popular_subreddits_driver.py". This script can be used as a recommended template example of how to produce a new script for a crawling job on this application.

The driver code starts with configuring an execution plan which defines the network name and submissions type that distinguishes this crawling job from others.

In addition, it is obligatory to define the different stages of this script in the "stages" array, which tells this driver code to jump or skip certain unnecessary technical stages under development, testing, or repair operations. An example of this execution plan is giving below

```
exec_plan = {
"run_1": {
"network_name": "Reddit_Most_Popular_Subreddits",
"submissions_type": "New",
"stages": ["crawling", "users_modelling", "activities_modelling", "statistics"]
}}
```

The driver script is executed by iterating over the configured runs in the execution plan, to perform the following 4 technical steps of this solution:

Step – 1: Crawling

This step is performed with the help of the Reddit crawling class located inside the script "classes/crawling/RedditCrawlClass.py.

Data provided through this class is written to the Mongo archive databases using the database connector class inside the script "MongoDBConnector.py", which is customized for the needs of this application and is located in the "classes/database_connectors" module within the source code.

Step – 2: Influence Graph Modelling

This step uses the class for influence graph modeling inside "classes/modelling/UserGraphModelling.py" to produce the desired influence graph, then store this graph in the respective neo4j database dedicated to storing influence graphs.

Step – 3: Activity Graph Modelling

Similar to step 2, this step uses the class for activity graph modeling in "classes/modelling/ActivityGraphModelling.py" to produce the desired activity graph, then store it in its respective neo4j database dedicated to storing activity graphs.

Step – 4: Generating Statistics and Monitoring Plots

Finally, the class called "Statistics" located inside "classes/statistics/Statistics.py" is used to produce the monitoring and statistical plots of this crawling job and store them on the file system of the running environment.

At every stage of executing the driver code, it is important to make sure that operations in the previous stage can provide valid data in a proper data structure, especially in step 1 where we need to make sure that the entities and attributes in the ER model shown in figure 2 are available and possible to process so they satisfy our established data structure for this application.

Also, steps 2 and 3 are both dependent on data stored in the Mongo archive databases after the process of crawling. And further on, step 4 depends on the results of all the previous steps to retrieve monitoring results about crawling and graph models.

In case there is a need to run only some selected stages, then simply provide the name of the stage inside the "stages" array included in the execution plan (exec_plan), and an "if statement" should do the job of skipping or executing the part of the driver code concerned with this stage.

The source code also includes a test driver that does not require any social network, and it is provided with dummy simulation data, in addition to another Reddit crawler that uses the same Reddit crawling class but to crawl the top 3 newest submissions from 5 selected subreddits.

All these examples of drivers follow the same skeleton and are named by adding the word "*_driver" to their script file name. They can also be set to run at regular time intervals by using a task scheduler on the operating system of the host machine of this application.

VI - G. Logging & Error Handling

Handling errors and exceptions is a vital measure for maintaining the health of any IT system. It is therefore important to log any errors or exceptions triggered under the execution of system procedures and operations.

This technical implementation implements error handling on a global execution level. i.e., at each script of a driver code or the server script of the web interface. Exceptions that might occur here or in any descendant code script should be caught by using a "try-except" control structure and then written to the "Logs" directory at the root of the source code. Errors are not stored in dedicated databases but on the file system of the host environment.

Logging is implemented according to the following guidelines to produce a hierarchical structure that helps isolate errors according to their origin:

1. The root directory of logged information is called "Logs" as mentioned above.

2. The children of the directory "Logs" can be numerous dated directories representing the actual date when an exception has occurred. For example, **"Logs/2021-05-20"**.

3. Inside any dated directory, multiple directories can be created for each crawling job by their network names. For example, **"Logs/2021-05-20/Reddit_Most_Popular_Subreddits"**.

4. Finally, a directory for each type of crawled submission is created under the directory of a network name, and the "error.log" file is stored at this path. For example, **"Logs/2021-05-20/Reddit_Most_Popular_Subreddits/New/error.log"**.

Also notice, that it is not always necessary to create the path of the inheriting directories needed for storing an "error.log" file, because if a crawling job executes without errors as expected, then there will be no need to create the directory path and neither an "error.log" file.

The creation and storage of error logs are implemented to automatically create the directory path and an "error.log" file when an exception occurs, then store the full thread of the captured exception in it.

A template showing the skeleton of the python snippet code that performs the logging mechanism in this technical solution is available in Appendix – A.

This code snippet takes into use the "logging" module which is a standard library module built in the python programming language.

The isolating hierarchical structure of the "Logs" directory helps developers and site administrators to keep track of errors when running more than one crawling job in the same hosting environment. Which results in a straightforward debugging experience.

VI - H. Data Protection

Protecting data is an integral part of most information systems, where it is important to maintain the integrity and confidentiality of system data to guarantee a high quality of the services provided to system users.

This technical solution is designed with an integrated database system for archiving data crawled from online social media, the database system runs on a Mongo DB Server Instance and offers the system administrator(s) the opportunity to reuse previously crawled data from online social media in a continuous timeline using the date of crawling.

Another important feature of having an archive database server is to serve as a backup. In case of any inconsistency in the activity and influence graphs stored on neo4j databases, a rerun of the driver script should read archived data from the relevant data batch and restore the consistency of graph data in the neo4j databases.

Data consistency in its structure and content is vital to guarantee a normal operation of this technical solution. Therefore, it is important to restrict access to every system database by defining a set of access modes and roles for reaching and modifying data in the system databases.

To help achieve better protection of data, it is recommended to use the least amount of entry points to the databases as done in this technical solution, where communication with the "Mongo" and "Neo4j" database servers is performed by using a corresponding customized connector class. A connector class can be imported and used when communication with databases is to be performed.

The classes "MongoDBConnector" and "GraphDBConnector" are provided in the "classes/database_connectors" module within the source code. They are used to establish a connection to and from the two "mongo" and "neo4j" database servers respectively.

These two classes are imported from the driver scripts for writing and reading access, and in the server script of the web interface for reading access only.

Because we wish to establish a restriction on modifying any data from the script of the web server, an input argument called "access_mode" is added to both database connector classes. This argument is checked internally in the created object of a database connector so writing and modifying methods cannot be called when the argument is having a value of "ReadOnly", which is giving to the database connector object in the script of the web server to prevent any potential attackers from taking advantage of the web interface to launch an attack that can modify or delete any data on the system.

Both database servers are recommended to run on a carefully protected environment on the infrastructure network, and if running on the same host of the web server, it is then important to block any access directly to the servers of system databases from outside the host machine.

Also, in addition to using strong passwords for any database servers, multiple roles of access can be implemented directly on these servers, which gives an extra line of defense against data corruption and manipulation from potentials attacks.

Although this system provides an archive and backup database server, it is also important to engage in controlling access according to the needs of the system administrator(s) and the purpose of this application. An example here is to consider restricting access to the web interface by implementing a proper authentication method like the HTTPS digest, two-factor, or one-time password authentications.

Digest access authentication is implemented in this technical solution because of its simplicity and satisfying level of security. More details about the implementation of digest authentication are given in the upcoming section on Securing UI[22] Routes with Digest Authentication.

In this aspect of the technical solution, multiple measures were introduced to offer better protection of data and system resources. These measures increase the reliability and security of this technical solution and make it suitable to run on a public server environment.

VI - I. Caching

Public clients using the web interface of this technical solution need to view all produced influence graphs and information about them in an efficient and fast way regardless of their size and complexity.

Because of the potentially big datasets, this system can handle, users can experience high latency when using some routes made available by the server that powers the web interface.

To mitigate the negative impact of latency on user experience, caching is implemented in the **"FLASK"** framework with the type "file system caching". Caching is set to work on routes that have the potential of struggling with latency after receiving HTTP(s) requests from clients.

An example of such routes is the one that reads training data from the mongo archive databases, then tune, train and evaluate the model of influence field classification, before providing the client with a full informative report about the performance of the text classifier used in the task of influence classification.

Caching is controlled from the ***".env"*** environment file by specifying the following 3 parameters:

1. **CACHE_ON**

   A "False" value of this parameter switches off the entire caching mechanism, and a "True" value does the opposite.

[22] User Interface

2. **CACHE_TIMEOUT**

   A default value of "0" tells the cache to produce cache records that never expire, and a positive integer indicates the number of seconds before expiration after the creation of a cache record.

3. **CACHE_DIR_PATH**

   Because we are using file system caching, we need to specify a place folder where cache records can be stored and retrieved from.

An open-source python module named "flask-caching" is installed and used to generate cache records after initializing a cache instance of this module, then mapping its configuration to the flask application instance as shown by the code snippet in Appendix - B.

As we see from the snippet code in Appendix - B, caching is mounted to work on the route ***"/topic_detection_model"*** with the cache configuration in the ***".env"*** file, specified by the system administrator(s). In addition, the query parameters from the user request are also included in caching, which makes it possible to produce a cache record for every possible influence classification model registered on the system, this is done by specifying ***"query_string=True"*** in the ***"cached()"*** decorator mounted on the declared route.

A cache record is generated and stored at the very first run of a route configured for caching. This means that a route will execute normally at its first run or after the expiration of its cache record, and then a cache record will be renewed with the configured expiration timeout in the future.

The entire cache storage of the system can be cleared using the protected route ***"/clear_cache"***. It is also possible to refresh cache records by specifying the unique parameters for a crawling batch using another protected route named ***"/refresh_cache"***.

The reason these two routes are protected by usernames and passwords is to protect against an attacker that can take advantage of continuously triggering these routes to clean and/or refresh cache records, which can result in consuming a lot of resources on the host machine and thus increasing the response latency due to the lack of cache records.

In addition to giving the system administrator(s) a protected route to clean and refresh cache records with, it is also a good idea to automate this behavior, so a creation or refresh of cache records can occur every time a new crawling job is performed by a driver script.

To help do this, a customized class called "CacheHandler" is included in the "classes/caching" module of the source code. An instance of this handler class can then be initialized and used to clear the specific cache records for a crawling batch by specifying the network name, types of submissions, and the crawling date. This cache handler will then ping each route configured to be cached to create a new cache record for it.

When using the approach of automated caching, it is recommended to set the timeout of cache records to forever by giving the environment variable called "CACHE_TIMEOUT" a value of 0, or a higher value than the time interval of the automated cache update. This way cache records would be refreshed when needed after a change has occurred on a certain batch of data or any other result of the running driver script(s).

Instead of running the operations of the route method every time the web server gets a request, caching will only run the route method the first time, then store its output result in a cache record to be used the next time a user requests the same route with the same URL parameters. This use of caching has a great value in increasing the availability and reliability of this technical solution.

The web interface has significantly benefited from using cache records to be able to provide a fast response to user requests, especially when running a public server for the web interface, where many identical requests are sent to this web server from different clients.

VI - J. Securing UI Routes with Digest Authentication

This technical solution is designed to deal with data that is generated by users of online social media and belongs to them by legal regulations in multiple countries. This increases the responsibility of this solution to protect the integrity and confidentiality of the user's data. Choosing a secure and suitable method for user credentials is therefore important to guarantee proper access control to system data and services.

Many methods of user access authentication can be used to secure the published routes on the web interface, the most basic method is the "HTTP basic authentication", which is a non-secure method due to its use of the easily reversible base64 hashing function, and its propagation of user credentials in plain text between client and server. And although it is possible to rely on the encryption of HTTPS/SSL to perform this basic authentication, the method of basic authentication can still be vulnerable to replay and phishing attacks.

A similar but more improved method of access authentication is the "digest access authentication", which uses the secure md5 hashing function, including nonce values to prevent replay attacks **[20]**.

A practical demonstration of the security of both basic and digest access authentication using the Wireshark packet sniffer is performed in the book titled "RESTful Java web services security: secure your RESTful applications against common vulnerabilities" by R. Enriquez and A. C. Salazar **[16]**.

Based on the previous evaluation of access control methods, digest access authentication is chosen in this solution to protect access to any route on the web interface.

The configured usernames and passwords are to be specified by the solution administrator(s) using two environment variables in the ***".env"*** file, where user credentials are separated by commas and placed at the corresponding positions to each other as follows in the example below

```
ADMIN_USERNAMES=first_username,second_username
ADMIN_PASSWORDS=first_password,second_password
```

These credentials are imported upon the start of the UI server and used to configure credentials after initializing an object of the ***"HTTPDigestAuth"*** class provided by the open-source python library ***"flask_httpauth"***.

To protect a route on the UI server, simply add the decorator ***"@auth.login_required"*** above the route method you wish to protect.

An example of such protection is implemented on the caching routes that enable a site administrator to clear or refresh the cache records on the system without having to log into the actual hosting machine. Restricting access to these cache routes is important as an attacker can rerun multiple requests to this route to clear cache records more often than needed, causing long delays in responding to service requests from system clients and in the worst case, a denial-of-service incident.

To add an extra level of security, it is also highly recommended to run any public site of the UI web server on the HTTPS/SSL protocol, and implement a URL redirect rule that forces any received HTTP requests to convert into using HTTP(s) to guarantee both the identity of the site server and that HTTPS is used for stronger encryption of the exchanged credentials between client and server.

More details about user authentication **[17]**, different cryptographic hash functions **[18],** and transport-level security **[19]** are available in the book titled "Cryptography and network security: principles and practice" **[17]-[19]**. This book has provided enormous help in the process of securing this technical implementation and its web interface.

In the published demonstration of this solution, we are only crawling publicly available data from Reddit, and this data is covered with the anonymity that Reddit offers its users. Because of this, routes to view graphs and many other analysis tools on the web UI are open to the public on the worldwide internet.

However, if this system is to be used for dealing with other more confidential data with little anonymity, it is recommended to protect all routes in the UI by using the provided flask decorator for the digest user authentication or consider implementing two-factor or one-time-password authentication for more security and access control.

VI - K. Supplementary Tools & Features

The source code of this technical solution is equipped with multiple supplementary tools and classes to provide additional helping features. The most important of such technical features are briefly described in the following parts of this section.

- **The Crawling Runtime Register**

This feature is implemented in the "RuntimeRegister" class, which defines a dictionary structure for monitoring how long it took the crawling phase to receive and process the requested data from the remote endpoint of an online social media.

This time register class is used both in the implemented Reddit crawling class and the test dummy crawler, it can also be used by any future crawling classes.

The data provided from this register is then stored by the driver code in the **"admin"** database of the Mongo DB Server under a collection named **"crawling_runtime_register"**. Every entry in this register has a timestamp and is marked with the unique key parameters of the crawling job specified in the driver code.

This register is also used in the **"statistics"** module to retrieve information about how much time it took to crawl information about the groups, submissions, comments, and training data for text classification.

Figure 21 below shows an example plot of runtime measurements of a crawled batch of data. Daily plots that show the runtimes of crawled batches are accessible to public users of this system via the "Statistics" tab on the web interface.

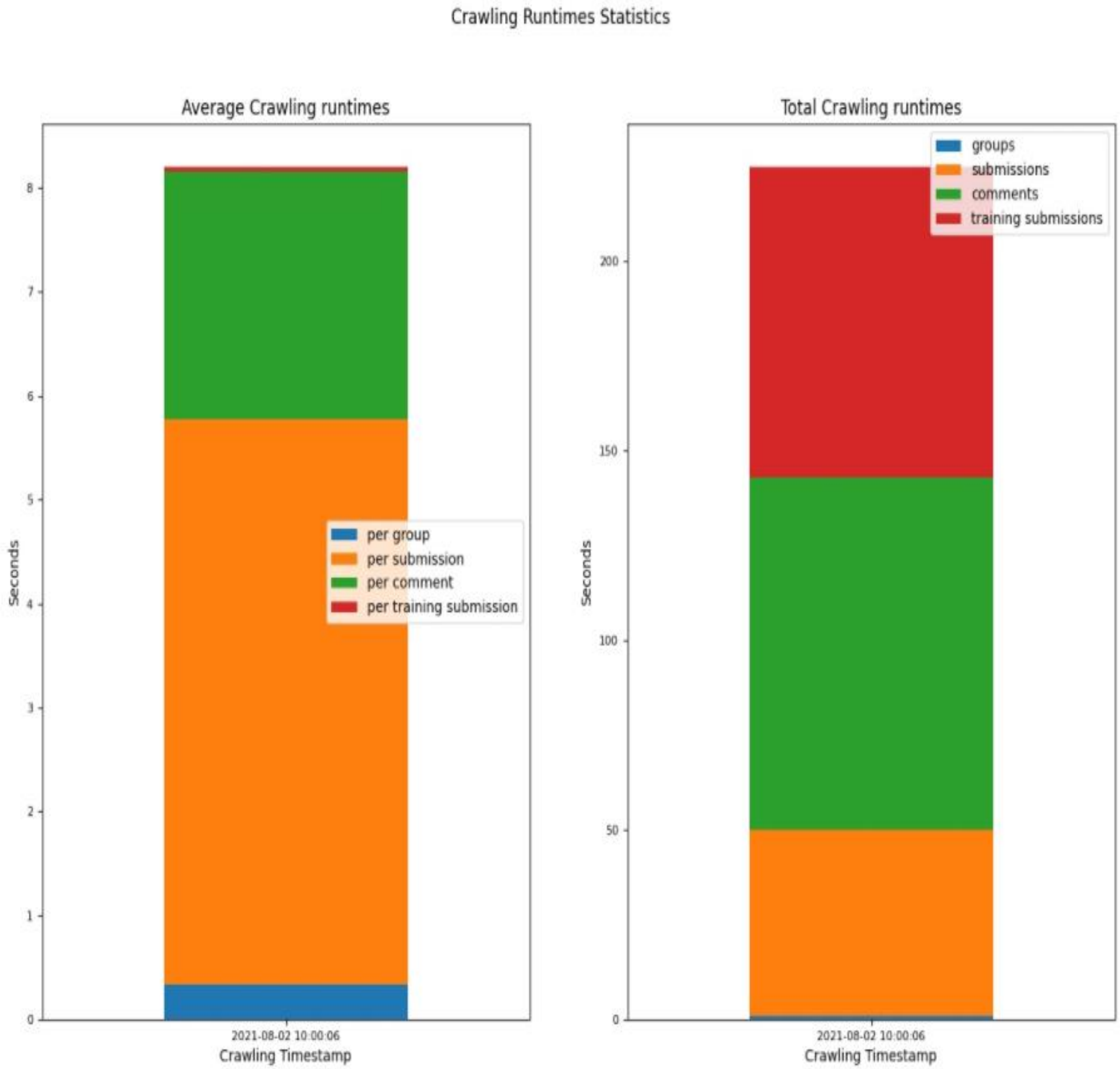

Figure 21, An example of a stacked plot showing the runtimes it took to crawl a batch of data from Reddit. This batch includes the top 3 newest submissions on the top 3 most popular subreddits, crawled on the 2nd of August 2021, 10 AM UTC.

- **Database Connectors**

Another important feature is to isolate the communication with the services of system databases, i.e., **"Mongo"** and **"Neo4J"** databases, by using dedicated classes that include desired queries and methods for each of them, which can be imported from the **"classes/database_connectors"** module when needed in the operations of this application.

This simplifies the processes of managing a future change in the syntax of database query language and communicating with the database, along with a better overview on access type and control, because we ensure that we only have one endpoint to reach the data in a certain database.

- **JavaScript Graph Visualization**

To enable public users to access the produced graphs of this system, a dynamic and browser-based visualization library called **"vis.js"**, this library uses the JavaScript language to visualize the activity and influence graphs on the client web browser.

Also, it helps showing any requested segment of these influence graphs on the web interface, such as showing the shortest influence path between the nodes of two distant persons in the graph.

- **Bootstrap**

The Bootstrap JavaScript library is used to build a responsive web interface and offer a better design of the used HTML[23] elements in the web interface of this technical solution.

- **JQuery**

This is a popular feature-rich JavaScript library that gives the front-end of this application a more summarized syntax and works together with both **"vis.js"** and **"Bootstrap"** libraries for better overall functionality.

- **Docker**

The entire application of this technical solution is packaged in multiple Docker containers, where the database servers and the web interface run each in its isolated environment. These environments are independent of the operating system of the host machine.

An advantage of using the Docker technology is the high automation in setting up and running the required resources of an application, which is noticeable when setting up a development or production environment of this technical solution using the Docker instructions included in the "README.md" file within the source code of this technical solution. A single command will instruct the docker engine to read the project configuration from the provided **".env"** file, download the required docker images of database services and set up their containers.

Also, a docker image with local code volume is constructed for this application with a python interpreter installed on it. And at last, docker spins up the web server for the user interface.

Docker also allows us to run multiple instances of the same application on the same machine, which is good for many reasons, such as when there is a need to distribute the application over multiple machines or having a series of backup environments in case things goes wrong in a current production environment.

All the advantages above highlight the role Docker technology plays in increasing the reliability and availability of this solution.

- **Removing Data on File System & The Databases of "Mongo" and "Neo4j"**

Developers and administrators of this solution might need to clear data from a certain database, or manually remove plots and cache records on the file system under debugging in development and testing environments. Or in production, where there might be a need to delete certain data batches before a certain date due to privacy laws and regulations such as the newly introduced rules of GDBR[24].

To achieve this task, the source code of this technical solution is equipped with a configurable script called **"remove_data.py"** which can be used manually or via a task scheduler on the hosting machine.

[23] HyperText Markup Language.

[24] General Data Protection Regulations.

VI - L. The Published Demonstration Site – https://smia.uis.no

An on-premises production environment of this technical solution was set up for demonstration and testing purposes. The setup uses the operating system "Microsoft Windows Server 2019 Standard", and many of its built-in features such as the IIS[25] webserver and the Windows Task Scheduler.

The following screenshot in figure 22 is taken from the main manager window of the IIS installation. It highlights the setup and some useful modules and features built-in with IIS.

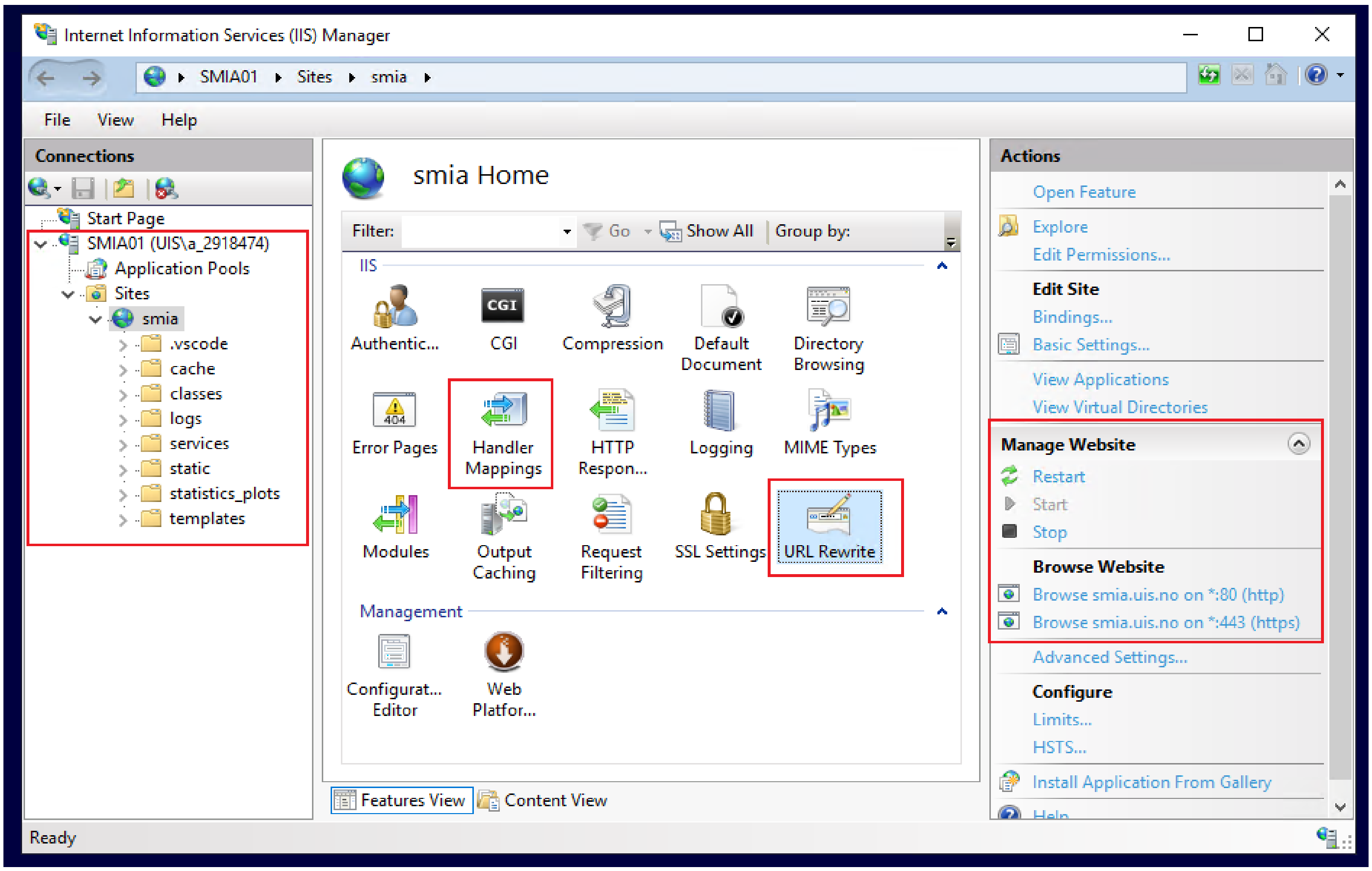

Figure 22, A screenshot of the main manager window of the IIS setup for the demonstration site of this technical solution.

Three automatic scheduled jobs were created to run the two driver scripts for downloading daily batches of data from Reddit. Along with the test driver script built on a small dataset of dummy simulation data.

The first driver script crawls the top 3 newest submissions inside the top 3 most popular subreddits. And the second driver script crawls 5 selected subreddits, each associated with a certain influence field.

The crawling is scheduled to run each day at 12 PM. Using the Windows Task Scheduler shown in the screenshot of figure 23 below.

[25] Internet Information Services.

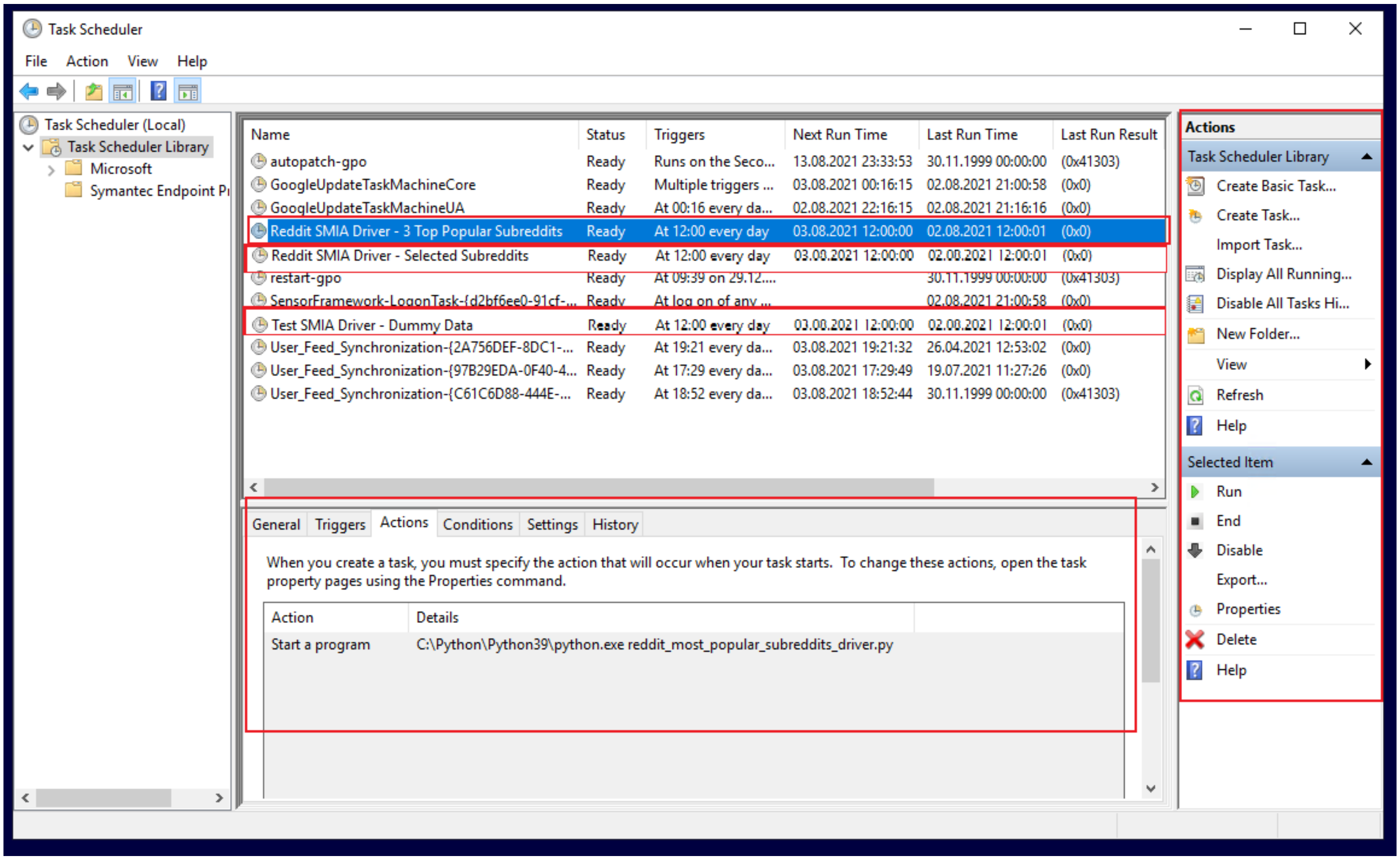

Figure 23, A screenshot of the setup on the Windows Task Scheduler for performing automated crawling of Reddit and dummy simulation data.

VI - M. Future Improvements of The Technical Implementation

Many parts of this technical solution are subject to continuous improvement to guarantee that it achieves its potential in being an effective and intuitive tool for continuous analysis of online social media. In this section, further details about possible future improvements of the technical solution are briefly described.

To begin with, we can redesign the crawling process to implement concurrency or distributed computing, meaning that each type of entities such as submissions and comments can be crawled and processed simultaneously. Which has the potential of reducing the amount of time it takes to crawl big data batches from online social media.

However, it is important to ensure the consistency of data as it should apply to the ER model of this project shown in figure 2 or a modified future version of it. Crawling entities at the same time without any consistency control might lead to a series of errors caused by deficient data like having some comments that should belong to a submission, where this submission is nowhere to be found in the submissions database.

Concurrency and distributed computing can also help the algorithms of graph modeling to reduce their running time and being able to handle more data at the same time.

In addition, the previously highlighted improvements of the influence graph model at the end of Part V in this research can be implemented in this technical solution to improve its future capabilities.

Furthermore, we can add a better UI visual alternative for model monitoring and viewing of statistics about each model, such alternatives can be to use a suitable JavaScript library for plotting data directly on the client-side of the UI instead of using the **"matplotlib"** library on the server-side.

Also, the front-end of the web interface does not have to use the **“bootstrap”** library, and it can be self-designed, or it can use any other desired JavaScript library to provide a better user experience in the future.

The creation of driver code scripts can also be automated so that data can be targeted and scheduled to run periodically using the web interface of this technical solution. This might propose a challenge regarding resource management, access control, and security issues, but many similar projects have lately been successful with these challenges with the help of cloud computing.

Furthermore, error handling and logging mechanisms can be improved by taking many measures such as storing the errors and different states of this technical solution into a database. Log messages can then be looked up and analyzed afterward to give the administrator(s) a report about the health and performance of the system.

The security of this technical solution is highly important and needs to be re-evaluated continuously, where it might be a good idea to consider using two-factor, 3-party, or one-time password authentication.

It is also important to avoid any unnecessary complexity of the security system as it might lead to hidden holes that can be misused to harm this technical solution. A balance between complexity and simplicity is often the right choice when securing most IT systems.

Finally, we can consider improving the caching system of the web interface by implementing another type of caching than the “filesystem” cache, such as the “Redis” cache which stores cache records in a dedicated database and has more features to be used for optimal cache performance.

It is also possible to design a new cache system that is customized to the need of this technical solution.

In addition, we can implement the possibility to create, remove or renew a cache record for a particular URL on the web interface, because in the current cache implementation, all cache records of a particular crawling batch are wiped out and a new set of records is created for these routes. This improvement can give the developers and administrators of this solution a more flexible tool for managing cache records.

We have hereby discussed some potential improvements that can help in increasing the reliability, availability, and security of this technical solution. It is also very important to evaluate these and other possible improvements in the future according to the gained outcome value of each improvement, and not to forget the importance of keeping a balance between complexity and ease of maintenance to achieve better performance with improved features of this technical solution.

CONCLUSION

We started the project of this research by establishing a ground data structure that fits into an entity-relationship model, then based on this model we defined a three-step algorithm for detecting and scoring influence between users of online social media; first by constructing the activity thread tree and activity graph, then transforming this activity graph into an influence graph.

The influence graph viewed users of social media as different nodes and visualized the influence flow between them as directed edges indicating that the person located at the beginning of an edge is the influencer, and the person being influenced is located at the target of an influence edge.

Those edges have also been scored by three different scoring techniques based on the number of interactions, upvotes, and activity rate. Each of these scoring techniques contributes to understanding the full picture of influence between users of online social media, which is why we combined the 3 scoring techniques of influence in a simple mathematical operation of addition and gained a wider range of score values, and higher variation between scores of different detected influences, such effect is desired for being able to distinguish and focus on influence edges having a significantly high score.

Furthermore, a text classifier was tuned and evaluated, then trained to predict the field of influence based on the text of the source- and target activities in the activity graph together with the submission text. This text classifier was integrated into the algorithm of building the influence graph to classify influence edges between influencers, which shows the possible field(s) of influence.

The process of text classification was designed to dynamically tune a text classifier for every daily crawled batch of data, providing the text classifier with an updated training dataset periodically.

The evaluation results of tuning this text classifier averaged to nearly 0.746 using the more reliable F1-score measure, which is considered a good score after noticing the natural overlapping between the targeted influence fields in the training dataset. Also, we had a classification problem that requires a mixture of supervised and unsupervised machine learning, and we only have a rough idea of what topic a certain text might belong to based on the kind of subreddit it was posted on.

Centrality measures such as the outdegree, betweenness, and HITS were calculated on the produced influence graph to compare the results with the individual influence edges in the same graph and have a better understanding of the importance of each influencer in the graph from a score-independent view.

The evaluation results showed a significant similarity between ranking by using the simple outdegree measure and the hub score from the HITS centrality measure, these two ranking systems had the advantage of extracting people with high direct influence on their neighbors. While the betweenness measure was useful in detecting influencers who help spread influence by laying on as many shortest influence paths as possible. On the other hand, the authority score from the HITS measure was effective in both determining influencers with high ability to participate in an influence transfer across the graph, and it can also reveal some people who are most likely to be influenced by others in the influence graph.

To apply these learned theoretical measures and newly developed algorithms to real life, a technical system was developed during this project. This system was designed to crawl and process data from a remote online social media platform on a periodic daily basis, then use this data to detect, score and classify influence between active users on a targeted group or segment of an online social media platform.

All this information was then visualized in an influence graph that is to be publicly available using a web interface.

This web interface also allows for retrieval of more information about each daily influence graph such as the crawling runtimes, topic shares in influence field classification, then histograms and boxplots of different influence score distributions, and not to forget the useful centrality rankings of user importance and the possibility to view the tuning results of each built text classifier.

The web interface also offers a series of tools to filter the graph based on a user-specified scoring criterium, a field(s) of influence, or to find an influence path between two known influencers with the help of the efficient capabilities of the neo4j graph database technology.

The reliability and security of the technical solution were also evaluated carefully and multiple measures such as logging, caching, and Digest User Authentication were added to support system management and increase the availability and reliability of the system for both administrators and public users.

After putting all these theoretical and practical pieces together, we construct a big picture that satisfies the objective at the start of this research, which is summarized in having an information system that crawls data from online social media, then detect, score, and classify different influence types between online influencers to produce a series of informative, flexible, and constructive influence graphs on periodic time intervals.

ACKNOWLEDGMENT

This project has benefited greatly from numerous resources listed in the attached reference lists, and I would like to thank every contributor for his work and dedication. You are also welcomed to contact me as the author of this thesis for any clarification.

The technical solution of this research had a great benefit from various online development communities and online documentation, and although I as the author of this thesis have done my best to credit each contribution specifically in the following listings of references [30]-[57], some deficiencies might occur and please let me know if this is the case.

In the end, I would like to thank the IT department at the University of Stavanger for providing the infrastructure that allowed for publishing a demonstration website of the developed technical solution during this research at https://smia.uis.no.

APPENDIX – A. LOGGING TEMPLATE

- The following template of python code is used for logging errors and captured exceptions in the scripts of the drivers and the server of the web interface inside the technical solution of this research. This code snippet takes in use the “logging” module, which is a standard library module built-in the python programming language:

```
from datetime import date
import logging
try:
    exec_plan = {
        "run_1": {
            "network_name": "Test",
            "submissions_type": "New",
            "stages": ["crawling", "users_modelling", "activities_modelling"]
        }
    }
    today_date = date.today()
    for run_id, config in exec_plan.items():
        # Name of social network to be crawled
        network_name = config["network_name"]
        # Assuming we are crawling the newest submissions
        submissions_type = config["submissions_type"]
        # date of crawling job
        date = str(today_date)
        # Driver code
        # ...
except Exception as e:
    log_path = F"Logs/{date}/{network_name}/{submissions_type}/"
    # create logs file if not found
    if not os.path.exists(log_path):
        os.makedirs(log_path)
    # create error logger
    logging.basicConfig(filename=F'{log_path}/errors.log', level=logging.INFO)
    # log error
    logging.error(F'\nError: {ctime(time())}\n{str(e)}\n', exc_info=True)
```

Appendix – B. Caching Template

- The python code snippet below, shows how caching is implemented inside the flask server of the web interface in this application:

```
from flask_caching import Cache

# Setting up caching
if os.environ.get('CACHE_ON') == "True":
    cache_type = 'FileSystemCache'  # cache records stored on file system
else:
    cache_type = 'NullCache'  # no cache
cache_timeout = int(os.environ.get('CACHE_TIMEOUT'))
config = {
    'CACHE_TYPE': cache_type,
    'CACHE_DIR': os.environ.get('CACHE_DIR_PATH'),
    'CACHE_DEFAULT_TIMEOUT': cache_timeout
}
app.config.from_mapping(config)
cache = Cache(app)

@ app.route('/topic_detection_model')
@ cache.cached(timeout=cache_timeout, query_string=True)
def topic_detection_model():
    # This route trains and evaluates the text classification model then returns results to user
    # This route method takes a significant amount of time to return without caching
```

APPENDIX – C. OUTDEGREE RANKED NODES VS. TOTAL INFLUENCE SCORE IN EDGES

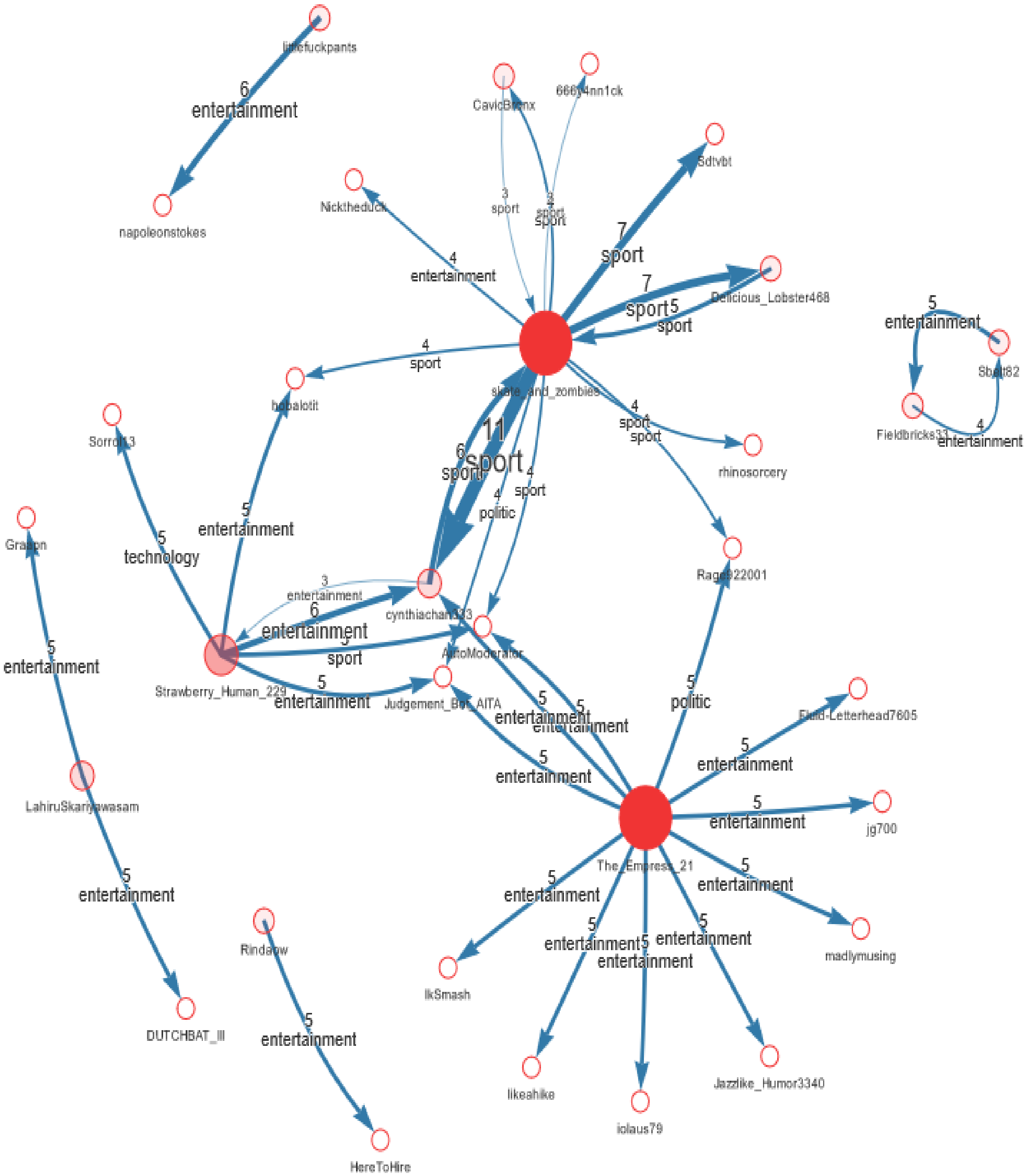

Figure 24, A visualization of an influence graph to compare between the use of node size and transparency to indicate the ranking of outdegree centrality, against the use of edge thickness to indicate the total influence score between network users in the same graph.

APPENDIX – D. BETWEENNESS RANKED NODES VS. TOTAL INFLUENCE SCORE IN EDGES

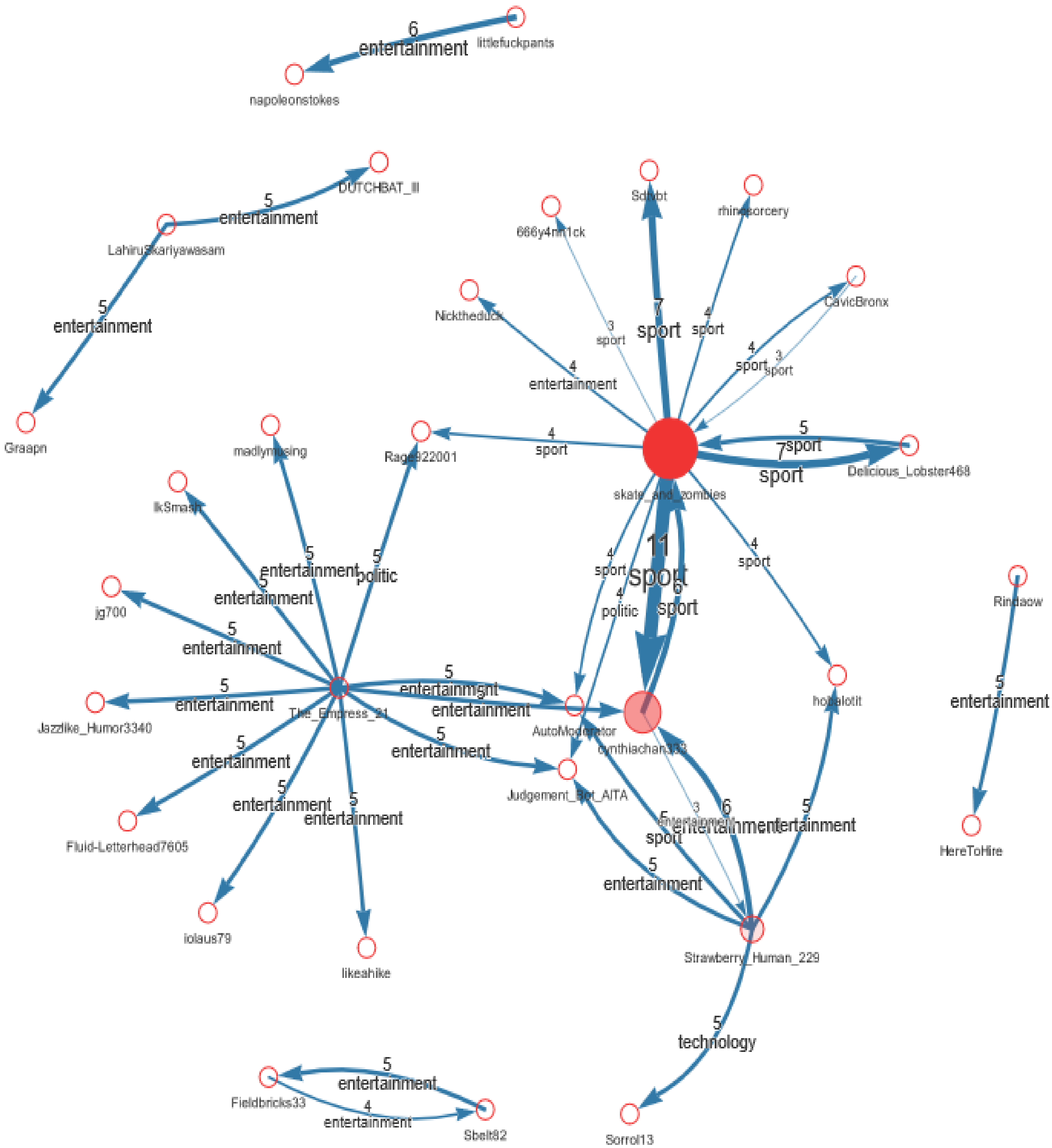

Figure 25, A visualization of an influence graph to compare between the use of node size and transparency to indicate the ranking of betweenness centrality, against the use of edge thickness to indicate the total influence score between network users in the same graph.

APPENDIX – E. HITS-AUTHORITY RANKED NODES VS. TOTAL INFLUENCE SCORE IN EDGES

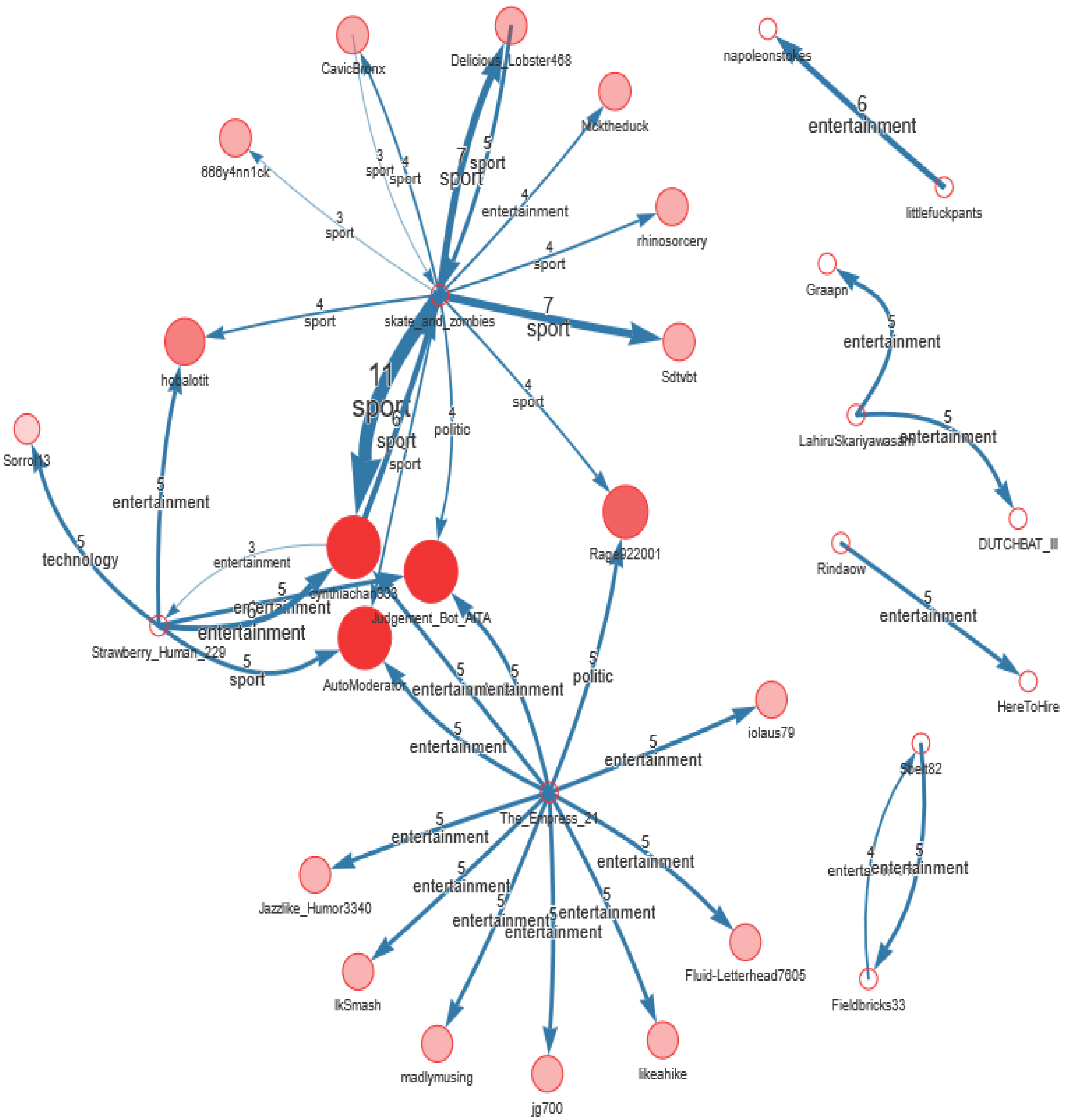

Figure 26, A visualization of an influence graph to compare between the use of node size and transparency to indicate the centrality ranking of HITS Authority, against the use of edge thickness to indicate the total influence score between network users in the same graph.

Appendix – F. HITS-Hub Ranked nodes vs. Total Influence Score in Edges

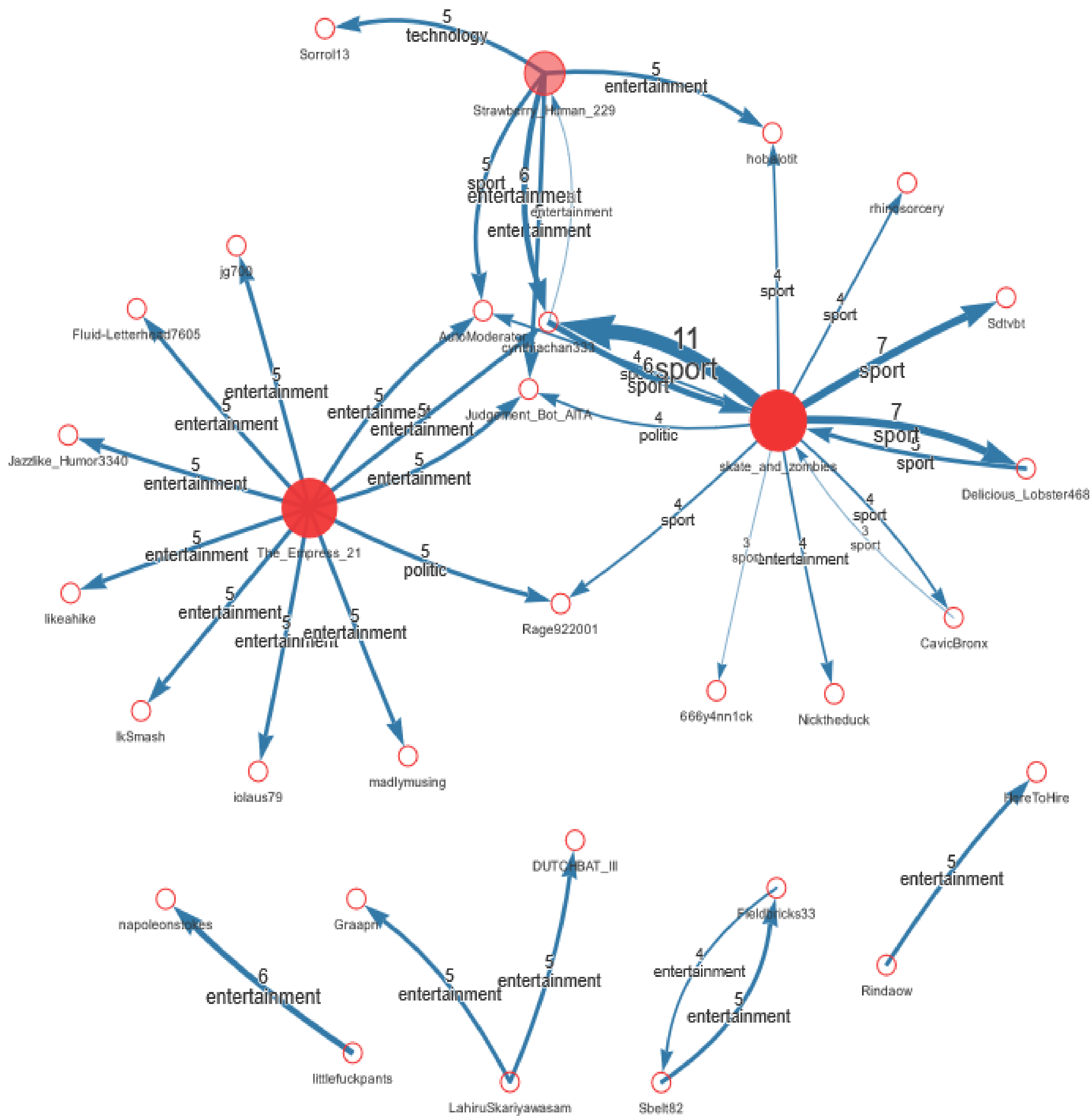

Figure 27, A visualization of an influence graph to compare between the use of node size and transparency to indicate the centrality ranking of HITS Hub, against the use of edge thickness to indicate the total influence score between network users in the same graph.

REFERENCES

[1] Y. Guo, J. Cao, and W. Lin, "Social Network Influence Analysis," 2019 6th International Conference on Dependable Systems and Their Applications (DSA), 2020, pp. 517-518, DOI: 10.1109/DSA.2019.00093. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[2] F. Wang, W. Jiang, G. Wang, and D. Xie, "Data-Driven Influence Learning in Social Networks," 2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC), 2017, pp. 1179-1185, DOI: 10.1109/ISPA/IUCC.2017.00177.

[3] Baumgartner, Jason & Zannettou, Savvas & Keegan, Brian & Squire, Megan & Blackburn, Jeremy. (2020). The Pushshift Reddit Dataset.

[4] Balakrishnan, Athira & Jones, Josette & Idicula, Sumam & Kulanthaivel, Anand & Zhang, Enming. (2021). Annotating and detecting topics in social media forum and modelling the annotation to derive directions-a case study. Journal of Big Data. 8. 10.1186/s40537-021-00429-7.

[5] T. Steinbaur, ‘‘Information and social analysis of Reddit,’’ inProc.TROYSTEINBAUER CS. UCSB. EDU, 2012, pp. 1–12. [Online]. Available: http://snap.stanford.edu/class/cs224w-2011/proj/tbower_Finalwriteup_v1.pdf.

[6] "About Statcounter GlobalStats" https://gs.statcounter.com/about (accessed July. 12, 2021).

[7] Statcounter GlobalStats, Social Media Stats Worldwide, Jan 2010 - Dec 2019. [Online]. accessed from https://gs.statcounter.com/social-media-stats#monthly-201001-201912-bar.

[8] "PRAW: The Python Reddit API Wrapper." https://praw.readthedocs.io/en/stable/ (accessed July. 12, 2021).

[9] "Working With Text Data" https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html (accessed July. 14, 2021).

[10] M. Schonlau, N. Guenther and I. Sucholutsky ‘‘Text mining with n-gram variables’’ The Stata Journal, (2017). 17, Number 4, pp. 866-881 [Online]. Available: https://journals.sagepub.com/doi/pdf/10.1177/1536867X1801700406 (accessed August. 06, 2021).

[11] ChengXiang Zhai and Sean Massung, “Chapter 6: Retrieval Models,” in Text data management and analysis: a practical introduction to information retrieval and text mining, 1st ed. San Rafael, California: Morgan & Claypool, 2016, pp. 87-128.

[12] J. Golbeck, “Analyzing Networks,” in Introduction to social media investigation: a hands-on approach, 1st ed. Amsterdam, Netherlands: Syngress, 2015, pp. 221-235.

[13] J. M. Kleinberg, “Authoritative Sources in a Hyperlinked Environment,” Journal of the ACM, 46, 5, 604, 1999. 10.1145/324133.324140.

[14] L. J. Grady and J. R. Polimeni, “Ranking,” in Discrete Calculus: Applied Analysis on Graphs for Computational Science, 1. Aufl. London: Springer Verlag London Limited, 2010, pp. 253-256.

[15] ChengXiang Zhai and Sean Massung, “PART III: Text Data Analysis,” in Text data management and analysis: a practical introduction to information retrieval and text mining, 1st ed. San Rafael, California: Morgan & Claypool, 2016, pp. 239-442.

[16] R. Enriquez and A. C. Salazar, “The Importance of Securing Web Services,” in RESTful Java web services security: secure your RESTful applications against common vulnerabilities, 1st ed. Birmingham, England: Packt Publishing, 2014, pp. 37-60.

[17] William Stallings, “User Authentication,” in Cryptography and network security: principles and practice, 6th ed. Harlow, England: Pearson, 2014, pp. 470-514.

[18] William Stallings, “Cryptographic Hash Functions,” in Cryptography and network security: principles and practice, 6th ed. Harlow, England: Pearson, 2014, pp. 333-374.

[19] William Stallings, “Transport-Level Security,” in Cryptography and network security: principles and practice, 6th ed. Harlow, England: Pearson, 2014, pp. 542-577.

[20] Klevjer, Henning & Jøsang, Audun & Varmedal, Kent. (2013). Extended HTTP Digest Access Authentication. 10.1007/978-3-642-37282-7_7.

[21] E. Zeitfogel, “Analysis of User Attention on Reddit,” Master thesis, Knowledge Technologies Institute, Graz University of Technology, Graz, 2014. [Online]. Available: https://diglib.tugraz.at/download.php?id=576a77f167aca&location=browse (accessed 20.07.2021).

[22] "FAIR Principles" https://www.go-fair.org/fair-principles/ (accessed August. 06, 2021).

[23] Ghosh, Anirudha & Sufian, A. & Sultana, Farhana & Chakrabarti, Amlan & De, Debashis. (2020). Fundamental Concepts of Convolutional Neural Network. 10.1007/978-3-030-32644-9_36.

[24] Hochreiter, Sepp & Schmidhuber, Jürgen. (1997). Long Short-term Memory. Neural computation. 9. 1735-80. 10.1162/neco.1997.9.8.1735.

[25] "Bidirectional LSTM" https://paperswithcode.com/method/bilstm (accessed July. 21, 2021).

[26] T. Wood. "F-Score." Website Title. https://deepai.org/machine-learning-glossary-and-terms/f-score (accessed 21.07.2021).

[27] C. Stafford. "Reddit" https://searchcio.techtarget.com/definition/Reddit (accessed August. 06, 2021).

[28] "3.1. Cross-validation: evaluating estimator performance" https://scikit-learn.org/stable/modules/cross_validation.html (accessed July. 30, 2021).

[29] "1.5. Stochastic Gradient Descent" https://scikit-learn.org/stable/modules/sgd.html (accessed July. 30, 2021).

[30] PyMoondra youtube channel. “Python Scripts - Scraping Reddit via API (PRAW).” youtube. https://www.youtube.com/watch?v=gIZJQmX-55U (accessed 20.07.2021).

[31] aldwinaldwin. “Mongo Create a user as admin for any database raise an error.” Stackexchange, Database Administrators. https://dba.stackexchange.com/questions/111727/mongo-create-a-user-as-admin-for-any-database-raise-an-error (accessed 20.07.2021).

[32] JoeyP. “Mongo user privilege to read and write to any database.” Stack Overflow, Questions. https://stackoverflow.com/questions/18823112/mongo-user-privilege-to-read-and-write-to-any-database (accessed 20.07.2021).

[33] Stampery Inc. “How to Enable Authentication on MongoDB.” Medium, mongoaudit. https://medium.com/mongoaudit/how-to-enable-authentication-on-mongodb-b9e8a924efac (accessed 20.07.2021).

[34] Jean-Christophe Chouinard. “Python Script Automation Using Task Scheduler (Windows).” JC Chouinard, Automation – Python - SEO. https://www.jcchouinard.com/python-automation-using-task-scheduler/ (accessed 20.07.2021).

[35] Bhargav Bachina. “Develop NodeJS REST API with MongoDB using Docker Compose.” medium, Bachina Labs. https://medium.com/bb-tutorials-and-thoughts/develop-nodejs-rest-api-with-mongodb-using-docker-compose-df9a37287aed (accessed 20.07.2021).

[36] λ.eranga. “How to Enable Authentication on MongoDB.” medium, Rahasak-Labs. https://medium.com/rahasak/enable-mongodb-authentication-with-docker-1b9f7d405a94 (accessed 20.07.2021).

[37] Thibaut. “How to install Neo4j with Docker-Compose?.” medium, thibaut-deveraux. https://thibaut-deveraux.medium.com/how-to-install-neo4j-with-docker-compose-36e3ba939af0 (accessed 20.07.2021).

[38] neverwalkaloner. “Docker so slow while installing pip requirements.” Stack Overflow, Questions. https://stackoverflow.com/questions/60086741/docker-so-slow-while-installing-pip-requirements (accessed 20.07.2021).

[39] Itamar Turner-Trauring. “Using Alpine can make Python Docker builds 50× slower.” pythonspeed, Docker. https://pythonspeed.com/articles/alpine-docker-python/ (accessed 20.07.2021).

[40] aljabear. “How to get the IP address of the docker host from inside a docker container.” Stack Overflow, Questions. https://stackoverflow.com/questions/22944631/how-to-get-the-ip-address-of-the-docker-host-from-inside-a-docker-container (accessed 20.07.2021).

[41] davidism & ThiefMaster. “Wait until database comes up.” GitHub, flask-sqlalchemy. https://github.com/pallets/flask-sqlalchemy/issues/630 (accessed 20.07.2021).

[42] community wiki. “Exploring Docker container's file system.” Stack Overflow, Questions. https://stackoverflow.com/questions/20813486/exploring-docker-containers-file-system (accessed 20.07.2021).

[43] Uday Hiwarale. "A beginner's guide to deploying a Docker application to production using Docker Compose." itnext, Docker: Docker Compose. https://itnext.io/a-beginners-guide-to-deploying-a-docker-application-to-production-using-docker-compose-de1feccd2893 (accessed 20.07.2021).

[44] Eyal Levin. "flask-cache memoize URL query string parameters as well." Stack Overflow, Questions. https://stackoverflow.com/questions/9413566/flask-cache-memoize-url-query-string-parameters-as-well (accessed 20.07.2021).

[45] Hafeezul Kareem. "Python – How to delete a file or folder?." mkyong, python. https://mkyong.com/python/python-how-to-delete-a-file-or-folder/ (accessed 20.07.2021).

[46] nikhilaggarwal3. "Create a directory in Python." GeeksforGeeks. https://www.geeksforgeeks.org/create-a-directory-in-python/ (accessed 20.07.2021).

[47] "Delete all files in a directory in Python." techiedelight. https://www.techiedelight.com/delete-all-files-directory-python/ (accessed 20.07.2021).

[48] ngShravil.py. "Insert many documents into empty collection, update if document with same key already exist for mongodb." Stackoverflow, Questions. https://stackoverflow.com/questions/62220625/insert-many-documents-into-empty-collection-update-if-document-with-same-key-al (accessed 21.07.2021).

[49] Shovalt. "UndefinedMetricWarning: F-score is ill-defined and being set to 0.0 in labels with no predicted samples." Stackoverflow, Questions. https://stackoverflow.com/questions/43162506/undefinedmetricwarning-f-score-is-ill-defined-and-being-set-to-0-0-in-labels-wi (accessed 21.07.2021).

[50] spectras. "Reading file using relative path in python project." Stackoverflow, Questions. https://stackoverflow.com/questions/40416072/reading-file-using-relative-path-in-python-project/40416154 (accessed 21.07.2021).

[51] Sheldore. "How do I add percentage in horizontal bar chart?." Stackoverflow, Questions. https://stackoverflow.com/questions/56741124/how-do-i-add-percentage-in-horizontal-bar-chart (accessed 21.07.2021).

[52] user9745009. "Flask Jinja2 - Parse JSON." Stackoverflow, Questions. https://stackoverflow.com/questions/50291923/flask-jinja2-parse-json?rq=1 (accessed 21.07.2021).

[53] Terentev Maksim. "Remove duplicate list in jinja." Stackoverflow, Questions. https://stackoverflow.com/questions/47648747/remove-duplicate-list-in-jinja (accessed 21.07.2021).

[54] Varun. "Python : How to copy a dictionary | Shallow Copy vs Deep Copy." thispointer. https://thispointer.com/python-how-to-copy-a-dictionary-shallow-copy-vs-deep-copy/ (accessed 21.07.2021).

[55] sh4nks. "__init__.py" github, flask-caching. https://github.com/sh4nks/flask-caching/blob/f11403fa0fb71e95cc73955c10b582911a613f39/flask_caching/__init__.py#L332-L339 (accessed 21.07.2021).

[56] sh4nks. "__init__.py" github, flask-caching. https://github.com/sh4nks/flask-caching/blob/f11403fa0fb71e95cc73955c10b582911a613f39/flask_caching/__init__.py#L390-L415 (accessed 21.07.2021).

[57] Smoe. "flask-cache memoize URL query string parameters as well." Stackoverflow, Questions. https://stackoverflow.com/questions/9413566/flask-cache-memoize-url-query-string-parameters-as-well (accessed 21.07.2021).

[58] "PRAW: The Python Reddit API Wrapper." PRAW Online Documentation. https://praw.readthedocs.io/en/stable/ (accessed 21.07.2021).

[59] "The Neo4j Graph Data Science Library Manual v1.6." neo4j Documentation. https://neo4j.com/docs/graph-data-science/current/ (accessed 21.07.2021).

[60] "Neo4j Developer Resources" neo4j Developer. https://neo4j.com/developer/ (accessed 21.07.2021).

[61] MongoDB, Inc 2008-present. "Enable Access Control." docs.mongodb.com, Security. https://docs.mongodb.com/manual/tutorial/enable-authentication/ (accessed 21.07.2021).

[62] MongoDB, Inc 2008-present. "The MongoDB 5.0 Manual." docs.mongodb.com. https://docs.mongodb.com/manual/ (accessed 21.07.2021).

[63] “PyMongo 3.12.0 Documentation" PyMongo Documentation. https://pymongo.readthedocs.io/en/stable/ (accessed 21.07.2021).

[64] “Python 3.9.6 documentation" Python Documentation. https://docs.python.org/3/ (accessed 21.07.2021).

[65] “Flask web development" Flask Documentation. https://flask.palletsprojects.com/en/2.0.x/ (accessed 21.07.2021).

[66] “Jinija template engine" Jinija Documentation. https://jinja.palletsprojects.com/en/3.0.x/ (accessed 21.07.2021).

[67] "JavaScript Tutorial" w3schools. https://www.w3schools.com/js/DEFAULT.asp (accessed 21.07.2021).

[68] "CSS Tutorial" w3schools. https://www.w3schools.com/css/ (accessed 21.07.2021).

[69] "HTML Tutorial" w3schools. https://www.w3schools.com/html/ (accessed 21.07.2021).

[70] “Jquery" Jquery API Documentation. https://api.jquery.com/ (accessed 21.07.2021).

[71] “vis.js" vis-network Documentation. https://visjs.github.io/vis-network/docs/network/ (accessed 21.07.2021).

[72] “vis.js" vis-network Examples. https://visjs.github.io/vis-network/examples/ (accessed 21.07.2021).

[73] “Docker Docs" Get started. https://docs.docker.com/get-started/ (accessed 21.07.2021).

[74] “Docker Docs" Docker Compose. https://docs.docker.com/get-started/ (accessed 21.07.2021).

[75] "Bootstrap 3 Tutorial" w3schools. https://www.w3schools.com/bootstrap/ (accessed 21.07.2021).

[76] “Bootstrap Documentation" getbootstrap.com. https://getbootstrap.com/docs/4.1/getting-started/introduction/ (accessed 21.07.2021).

[77] “flask-caching documentation" flask-caching. https://flask-caching.readthedocs.io/en/latest/ (accessed 21.07.2021).

[78] “Digest authentication example" flask-httpauth. https://flask-httpauth.readthedocs.io/en/latest/ (accessed 21.07.2021).

[79] “Custom Error Pages" Flask Error Handling. https://flask.palletsprojects.com/en/1.1.x/patterns/errorpages/ (accessed 21.07.2021).

[80] “Internet Information Services (IIS) for windows" IIS overview. https://www.iis.net/overview (accessed 21.07.2021).

[81] “Task Scheduler for developers" Windows Developer. https://docs.microsoft.com/en-us/windows/win32/taskschd/task-scheduler-start-page (accessed 21.07.2021).

[82] “NumPy v1.21 Manual” NumPy. https://numpy.org/doc/stable/ (accessed 06.08.2021).

[83] “Pandas” Pandas Guide. https://pandasguide.readthedocs.io/en/latest/ (accessed 06.08.2021).

[84] “matplotlib version 3.4.2” matplotlib. https://matplotlib.org/stable/contents.html (accessed 06.08.2021).