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	…… (Writer's signature)
Faculty supervisors: Antorweep Chakravorty Nikita Rajendra Karandikar	
Thesis title: Identifying and Classifying Social Influencers - Engagement Analysis	
Credits (ECTS): 30 Points	
Key words: online-social-media, influence-detection, influence-weight, influence-field-classification, text-classification, graph-centrality, graph-node-ranking, IT-system-architecture, logging, caching, digest-authentication, mongo-database-server, neo4j-graph-database-server, python, flask, javascript, html, css, jinja-templates, bootstrap, jquery, vis.js, git, windows-iis-server, docker.	Pages: 46 + enclosure: Source Code Stavanger, 10.08.2021

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 detecting, measuring, and classifying the strength and topic field of influence in the process. This project proposes a method for influence detection, then build 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 social media network.

Keywords: online-social-media, influence-detection, influence-weight, influence-field-classification, text-classification, graph-centrality, graph-node-ranking, IT-system-architecture, logging, caching, digest-authentication, mongo-database-server, neo4j-graph-database-server, python, flask, javascript, html, css, jinja-templates, bootstrap, jquery, vis.js, git, windows-iis-server, docker.

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 function and socializing no longer requires the physical presence of society members. As a result of the introduction of multiple social media platforms, people from all over the world can now engage in local, national, and global events, participating in society and expressing 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 social media platforms, our society has become almost totally dependent on such platforms, and most social events and happenings does not pass away from 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 on 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 which often increase the competition in economy and generate more values for companies and their surrounding societies.

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

I. INTRODUCTION

Data from social media has a 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.

The aim of this research is to establish a ground foundation for extracting information about user activities on social media and use such information to detect social influence between network users. Such foundation is desired to make up the core of a future technical solution that enables social media analyzers with little or no technical experience in data processing and visualization to perform social media analysis on regular periods with a continuous timeline.

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, and 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 with respect to direction of influence, and holding the score of influence strength along with its classified field of influence whether it is in sports, politics, or economy etc.

Following the previous effort, we evaluate the performance of the influence graph model and go through test results from both dummy and real-life data using crawled data 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 model of the influence graph by ensuring 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 and 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 of the upcoming research in a nutshell, but first let's take notes and learn from some interesting pre-attempts in studying 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, and a great amount of effort has been dedicated to studying social influence between users to better understand the behaviour of individuals for many purposes. Research of social influence takes different forms and vary in size and scope, while some researchers take on the very fundamentals of detecting social influence, others dive through it to reveal details such as a specific influence or hidden behaviour patterns on different levels. In this section, we explore some related work in the field of social activity on online social media and try to get an inspiration that helps direct the effort of this research in 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. 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 structures 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. [1]

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. In this paper, the process of influence diffusion is divided into two parts: the launcher (influence strength) and the receiver (influence threshold) which can generate an accurate and finer grained influence diffusion model according to this research. [2]

Furthermore, the researchers highlight the importance of having a solid criterium when scoring the strength and threshold properties of social influences. Another important acknowledgment is the difficulty and complexity associated with detecting influence relationships between users as a by-product of big datasets that usually include a considerable amount of noisy or less important datapoints, making it essential for any algorithm used in learning and testing the 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 spot the light on this initial aspect of gathering data from social media platforms. A significant research is one that 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

undirected 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 endpoint.

It also gives an excellent brief description of the FAIR data[1] 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", highlighting their strengths and weaknesses in a constructive manner.

II - D. Topic Detection in Socical 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 the purpose of classification between different topics where a certain user activity might fit in. Inspiration on possible solutions for this task can be obtained from a research about annotating and detecting topics in social media forum and modelling the annotation to derive directions 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 to give a good introduction of 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 the purpose of topic classification.

Another contribution of this research is the use of various deep learning algorithms to classify posted content such as CNN[2], LSTM[3] and BiLSTM[4], all in which enable the researchers to achieve a promising F1-score[5] 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 [4].

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

Determining which social media platform to crawl under testing and evaluation of a new modelling approach is important to produce a flexible influence model that can be used for analysis of as many social media platforms as possible, this is why 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 posts or 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 off 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 virtual platform of Reddit, this analysis helps explaining why Reddit should be used in evaluating the performance of an influence model and its ability to view the most influencing users in a social media platform. The reason for this is Steinbauer's detailed analysis on which subreddits seems to have the most of user's activity, and in addition his further construction of an example user-oritened influence graph that helps showing 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 modelling of an influence graph is the limitation of not using any other criteria than user interaction through comments, such as the upvote score or number of thread or descendant comments posted on other comments or submissions.

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 has been used for producing the model of the influence graph are not provided to the reader in satisfying details.

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

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, where market share data is obtained from statcounter.com which claims to base its statistics about digital market shares on a sample exceeding 10 billion pageviews per month. [6]

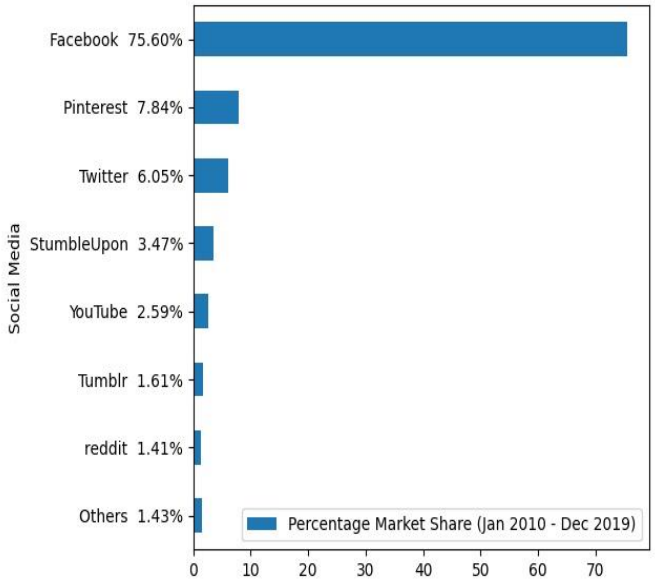

Figure 1, Worldwide market share of top popular social media platforms from the beginning of 2010 to the end of 2019 [7]

Although Facebook is the definite leading social media platform, it is still possible to observe a competition in popularity when looking at the next 6 platforms below Facebook, with Reddit having a popularity corresponding to all other social media platforms that are less popular than Reddit.

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

The market share of Reddit does nearly equal as the sum of at least 14 other social media platforms and all of these are reported to be below reddit in popularity including well established platforms such as LinkedIn and Instagram. This makes Reddit a suitable candidate to be used as a case study 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 tends 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 offers its users an opportunity to interact with each other in many aspects of socializing like professional and personal life combined. Reddit is considered as one those digital news platforms which is still gaining popularity and increasing in content since its launch in 2005.[6]

A user on Reddit can create or join a group, make a submission on any group and comment on any submission or comment made by other users. A user can join a group, but it is not obligatory 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 newspapers which is often divided into pages for multiple areas of concern such as politics, economy, or sports. 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 Reddit attracts users by their interest in topics and events in their social surroundings, while other social media often relays on the social affiliation of a future user. However, and on the other hand, 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 as common functionality increases the modelling flexibility to be used on other social media platforms in future analysis, and its separation of social environments in groups 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 to give 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 can also choose a username that can be used to identify them personally if they desire.

Another good reason for choosing Reddit as a study case is the highly developed endpoint crawling API which is very object-oriented and offers a wrapper library for the Python language.

[6] Reddit is a social news aggregation, web content rating, and discussion website [27].

This eliminates the bother of dealing with HTTP requests and latency issues, as all of this is taken care off in the background of the Python Reddit API Wrapper. The Wrapper is free to use but it requires a registration which once done offers no restrictions on how often Reddit is crawled, unlike crawling by adding ".json" to the URL which have its downsides such as 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. In other word the PRAW[7] 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.

- Accessibility:

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

- Interoperability:

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

- 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 present and future technologies.

A complete study that analysis 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 and 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. We shall than design our ground data structure to adapt for the common functionality between Reddit and the most popular social media platforms as we will go through in the upcoming section on the ground data structure.

IV. DEFINIING A GROUND DATA STRUCTURE

Flexibility of design is an important requirement of this research, as we aim for a future application of the influence modelling developed in this research on as many other social media platforms as possible, and although this might be difficult to achieve as a result of the wide variety of available social media and the different user-functionalities in them, we can still notice some common user functionalities between the most popular social media platforms such as LinkedIn, Facebook and Reddit, this common functionality is no accident, as these social media platforms most likely inspired from real life social interactions 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 for 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 real life society. Some people get influenced without any big 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 a social interaction. For this reason, we are going to look at active social influence where we would expect the person who get influenced to react by submitting an activity on the content of influence. 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 is able to 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.

[7] Python Reddit API Wrapper

The model can initially be based on four different entities a user can create and interact to; these entities are:

1. Network

   Which holds information about the crawled social media platform or a smaller segment of it, having this entity, makes it possible to study multiple social media platforms at the same time which increases the flexibility of design.

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 author and information about the location of a comment in the comment thread. And therefore, it has an additional feature which its ability to be a parent and/or a child of other comments. This means that comments can be modelled as a tree data structure that can grow unlimited.

The figure 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 entities explained earlier in this section, 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 that is a child descendant of the comment tree of a certain submission.

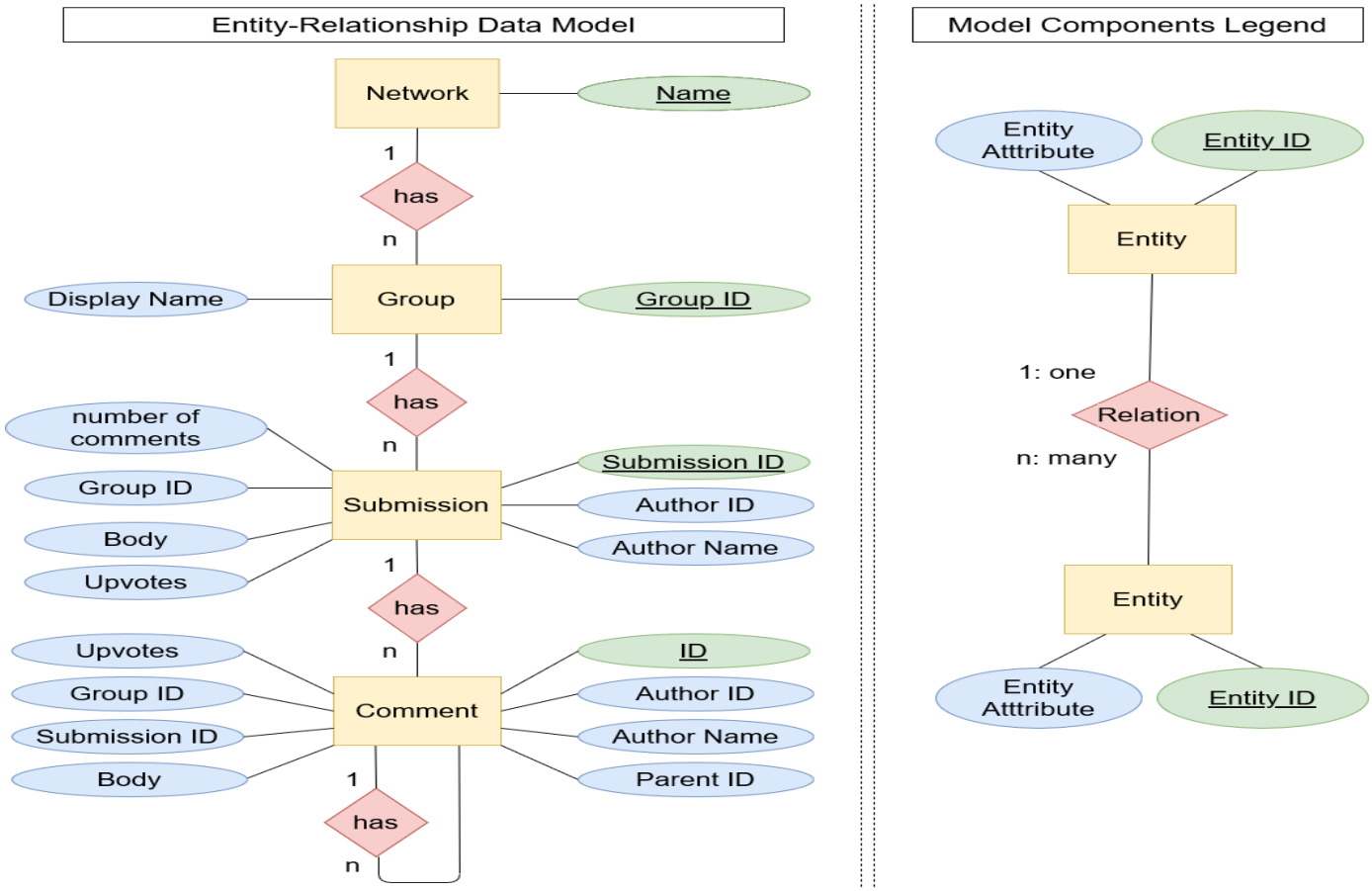

Figure 2, The Entity-Relationship model which represents the ground data structure of this research.

Most of the popular social media platforms contains the four identified entities in this ER-model[8], although they might have a different name, form, or purpose such as a company page on LinkedIn or a user profile on Facebook, but both can be treated as groups just like Reddit groups as well.

The attributes of entities open for more flexibility as we might have the need to extend or shrink our ER-model in the future i.e., by not including an upvote attribute or by adding a reaction attribute to submissions and comments. But also, attributes are generalized to match the very common details about of these entities in between the most popular social media platforms.

Now that we have a ground Entity-Relationship model to base our data structure on, we can proceed into discovering influences between users based of the interactions between them using these entities and relationship roles established in this section. In the next steps, we will elaborate the different stages of our influence modelling process for detecting and extracting social influence from online social media, the algorithms of the upcoming influence modelling are expected to have multiple dimensions for revealing the strength and types of social influence between different users online.

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 of detecting activity- and interaction-based social influence, 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 as a tree structure where network contains multiple groups that in turn contain a series of submissions which can contain multiple branches of comments, comments can have their own comments resulting in a tree that can grow endlessly as users add more comments on previous comments.

The figure 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 each single node in this thread represents an activity object that a certain user had performed.

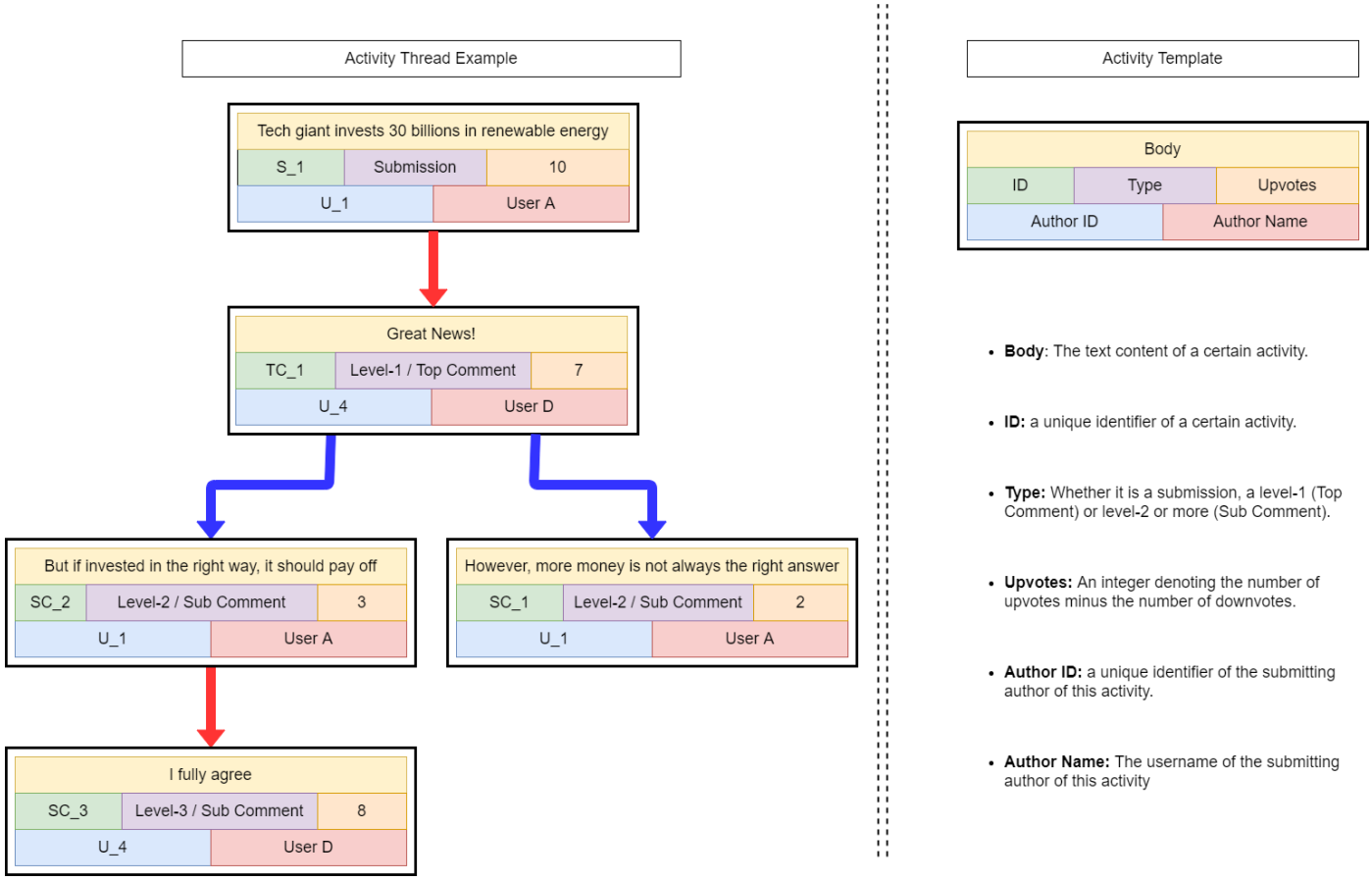

Figure 3, A set of submissions and comments is modelled as an activity tree by taking advantage of its natural thread hierarchy.

[8] Entity-Relationship model

In the activity thread drawn in figure 3 above, we have 2 participating authors, the 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 activity of "User A", and a blue arrow indicates an interaction from "User A" to the activity of "User D". In the next steps of influence modelling, we will investigate how we can use this activity thread to give birth to a weighted and directed activity-oritened 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 grow of the activity thread by defining comments based on their level in the activity tree to avoid any confusion when referring to different types of comments. As comments can have their own thread comments, we can establish a categorization of comments based on their level position in the thread hierarchy tree, where we can use two types of comments:

- Top Level Comments

  Which are comments made directly on a submission.

- Sub Level Comments

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

The distribution of comments between top comments and sub comment 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 each edge represents one interaction between 2 submitters where target and source nodes are represented by their authors, showing 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 users, so we can determine how strong a certain influence is in comparison to other influences in the influence graph topology. For this reason, we will give each influence edge a score that shows its strength.

Many techniques can be used to perform the scoring of influence; however, each technique have its strengths and weaknesses, and one way to reinforce the analysis system is by offering multiple scoring techniques to be used in analysis, to accomplish this, three different scoring techniques are used, each having some strengths and weaknesses depending on the use case of final influence analysis.

1. Interaction

This scoring technique measures how many times a studied user has reacted to activities performed on 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. But depending on the network, many users tend to interact one or two times in most cases, this is despite that the topic might be very interesting users who does interact that much. To accommodate such cases of low interaction influence, the following two scoring techniques can be applied in analysis.

2. Upvotes

By using the difference between upvotes or likes and downvotes or dislikes on a parent activity, we rely on the audience opinion on 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 from the author of this activity having this influence score.

This scoring method gives a democratic approach that enables us to know whether an influence activity is supported or downvoted by a group of interested audience. At the same time, it is important to notice that in some networks such as Facebook, it is not possible for users to downvote an activity, which leave us without knowing for sure whether the influence 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 downside of this upvotes-based scoring technique is that all child activities will get the same score, which can result in low differentiating effect between influence edges in the influence graph model.

3. Activity

The activity scoring technique scores influences based on how many edges of influence are found in the full branch of influence started from a parent activity.

Central people in society like politicians tends to influence the most people by triggering a great number of 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 technique 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 oritened calculation, and it does not take in account all descendant activities of the parent activity.

The figure 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 that 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.

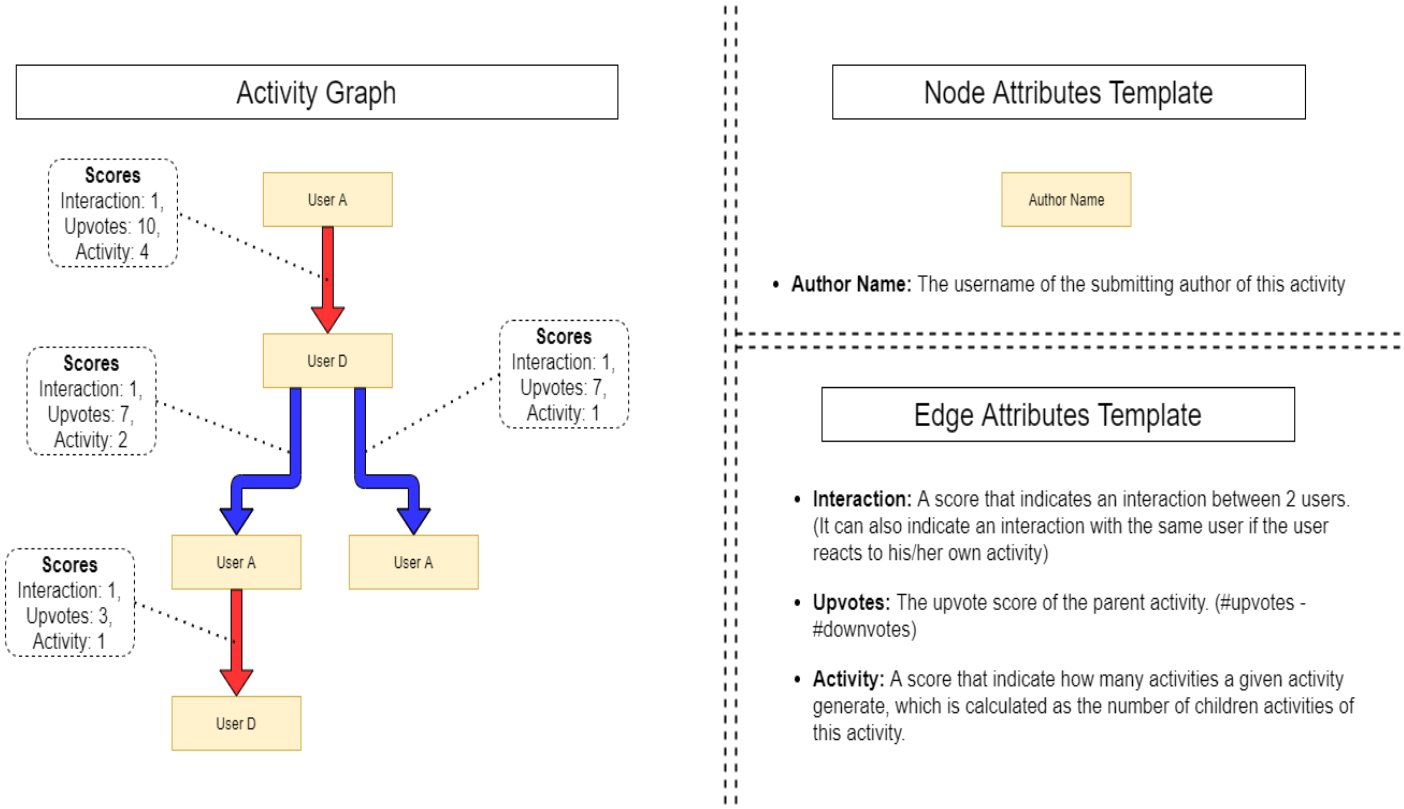

Figure 4, The activity graph is constructed by representing the nodes of activities using their authors, and applying the scoring techniques of interaction, upvotes and activity to score the individual influence edges.

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 oritened in the opposite direction. We can digest the structure of this directed and weighted activity-oritened 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 a person-oriented influence graph, where each node represents a unique author in this graph, and where edges are used to indicate a particular scored influence between two persons with respect to 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 steps:

1. Creating a person node for each unique author in the activity graph.
2. Listing all influence edges stretched from "User A" to "User D" in the activity graph.
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.

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 in activities submitted by "User D", while the upvotes score is the sum of upvotes values recorded on these unmerged edges in the activity-oritened graph and the same goes for the activity score.

Following the four steps above, a new person-oritened 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 resulted new influence graph visualizes the flow of influence between persons, how strong each influence is, and in which direction between them it is observed. An example of the previous four steps transformation from the activity graph in figure 4 to a person-oritened influence graph is presented in figure 5 below.

The example user-oritened influence graph below does have two users with two influence edges between them; according to the interaction score, both users does have equal influence in 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 technique.

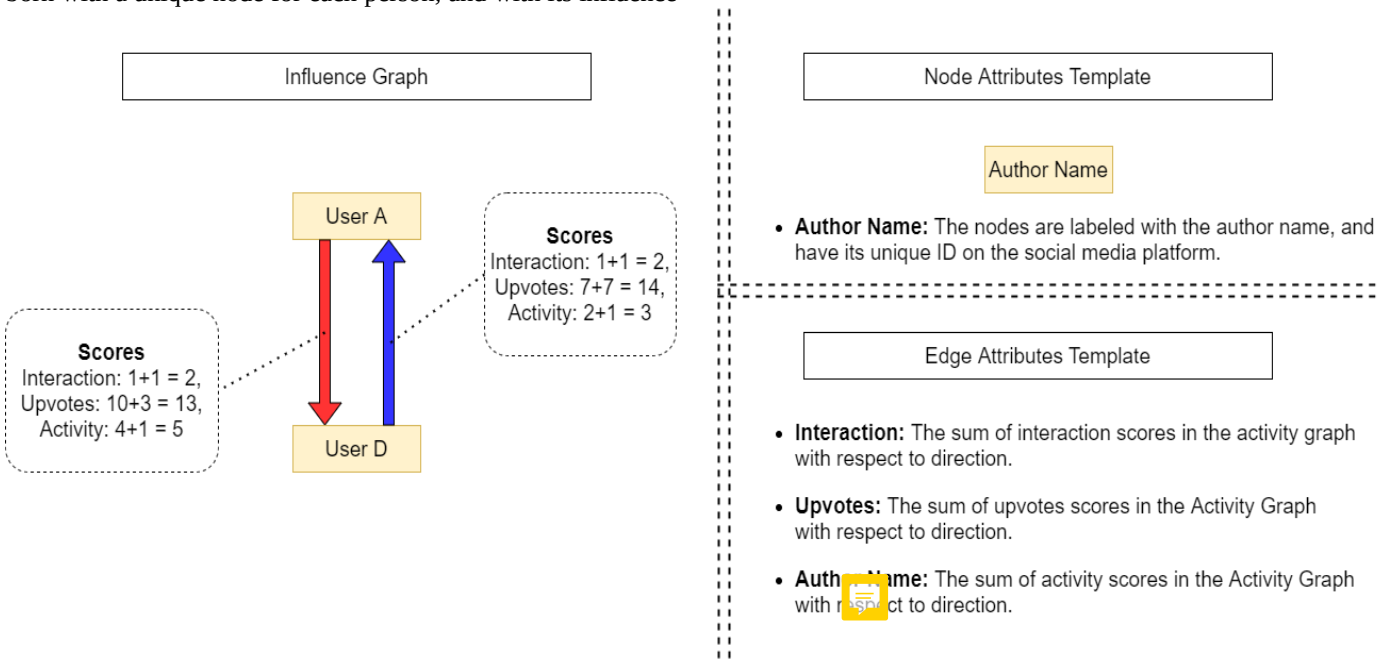

Figure 5, The user-oriented influence graph contains a unique node for each person, and the edges of this graph indicate the weights and directions of influence between the persons in this influence graph.

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 influenced person.

We are primally 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 score values obtained by using different influence scoring techniques that were detailed previously in this research.

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

The following figure shows 3 box plots and 3 histograms that shows the score value distribution of each scoring measure recorded in edges of the person-oritened influence graph.

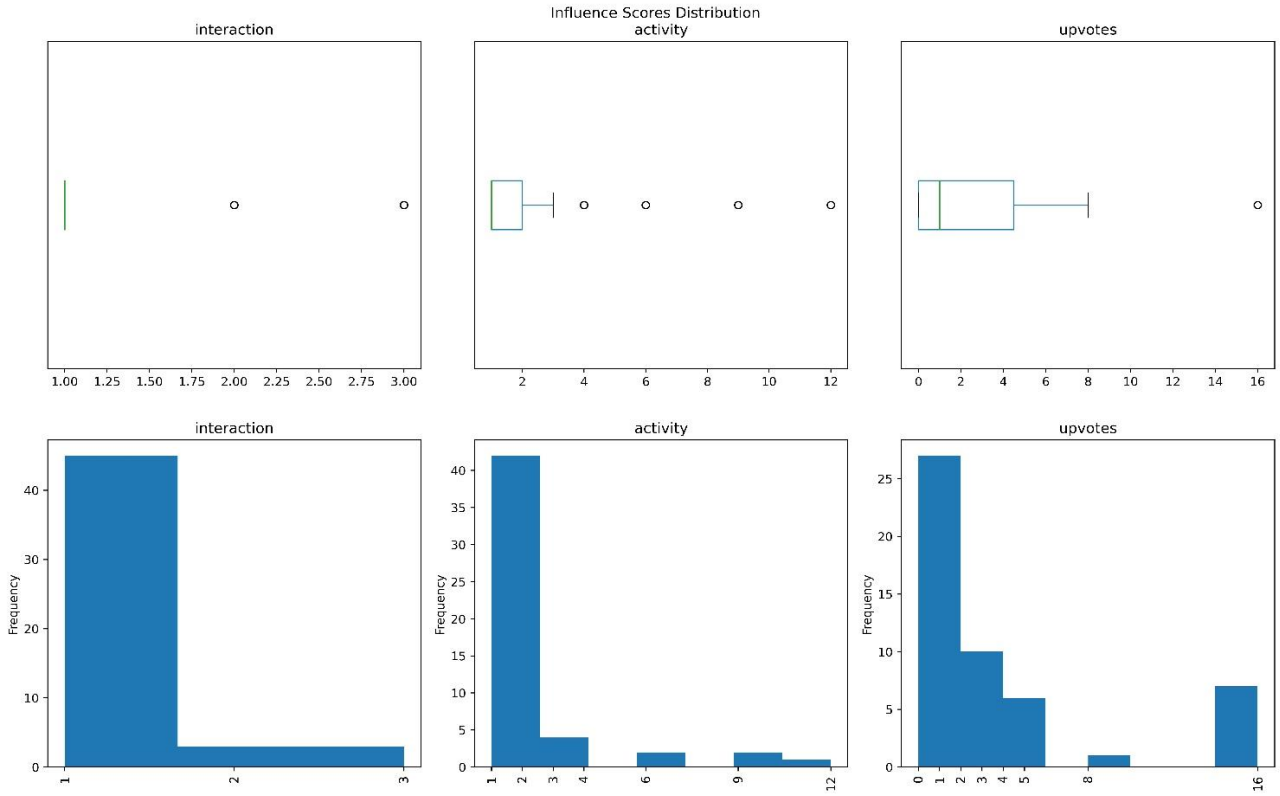

Figure 6, The single score distributions recorded in the edges of the influence graph, constructed on a dataset containing the top 3 newest submissions posted inside the top 3 most popular subreddits on the 12th of July 2021 between 12 to 12:30 O'clock.

Examining the distribution of interaction-based score values in the left box- and histogram plots in figure 6 above, gives a clear observation of low variance between score values which reduce the differentiating capability of the interaction scoring technique. However, a relatively good number of influences were successfully detected from the crawled dataset.

Moving on to the examination of activity-based score values in the middle of figure 6 above, we notice a slight decrease in the number of influences having the lowest score in comparison with the interaction-based score values on the left side. In addition, there is a noticeable extend in variance as the range of recorded values has increased from 1 to 3 in the interaction-based score values to a range between 1 to 12 using the activity score values. Four outliers on the right side of the box plot of activity scores also tells us that this scoring measure has more potential to differentiate between influences and present the detected influences that are having more significance than others.

Finally, we examine the upvotes-based score values to discover a higher achieved variance and a smaller number of outliers in the distribution of obtained score values. This result shows us the effectiveness of using the upvotes-based scoring technique for differentiating between detected influences.

Because each scoring technique might have its strengths and weaknesses in accordance with the purpose and motivation of the performed analysis, an analyzer might also wish to combine multiple scoring techniques and look on the distribution outcome of score values. In this research, the single scoring techniques are combined by performing an addition operation on all possible combination of the 3 scoring techniques in each edge in the influence graph. This results in the following four new scoring techniques:

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

Figure 7 below does plot the distribution of score values obtained from the double combined scoring techniques. An increase of variance is observed on all double combinations of scoring measures, especially when combining the activity and upvotes scoring techniques, which gives the highest achieved variance and therefore the best capability of differentiating between detected influences based on their scores, and independent of the purpose and motivation of analysis.

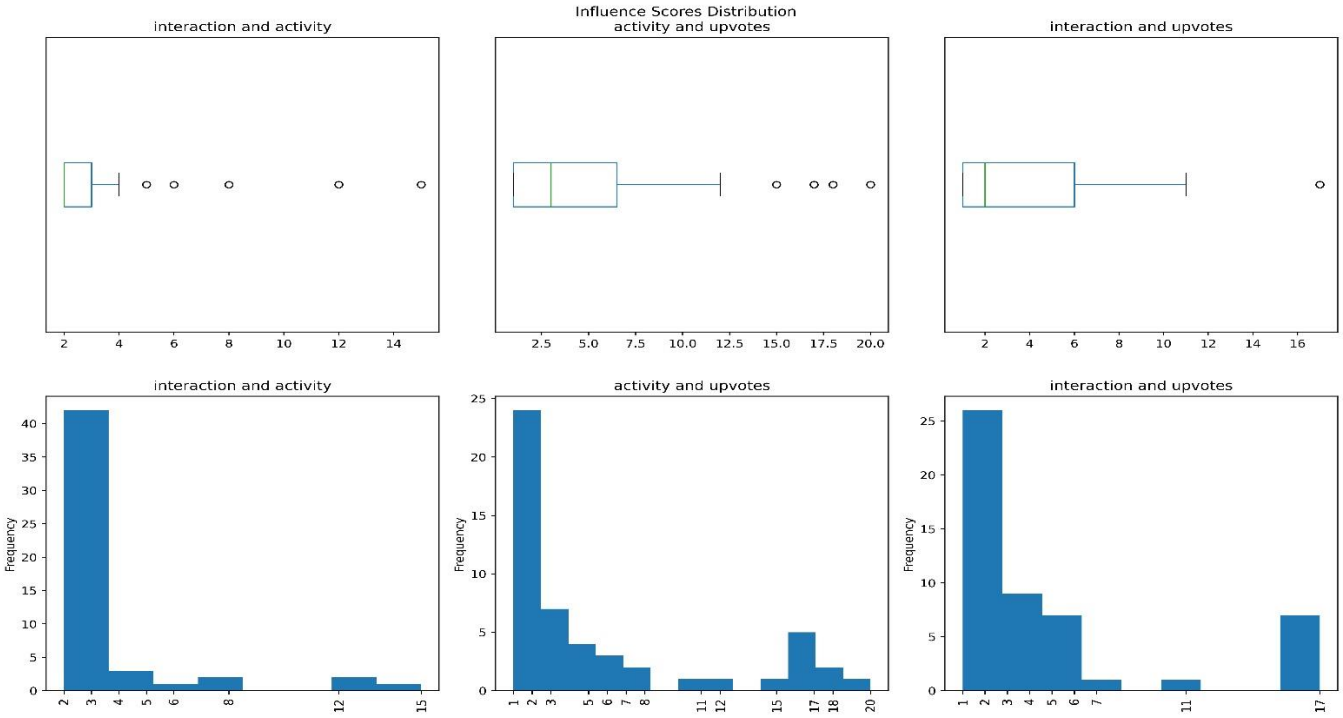

Figure 7, The double score distributions recorded in the edges of the influence graph, constructed on a dataset containing the top 3 newest submissions posted inside the top 3 most popular subreddits on the 12th of July 2021 between 12 to 12:30 O'clock.

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

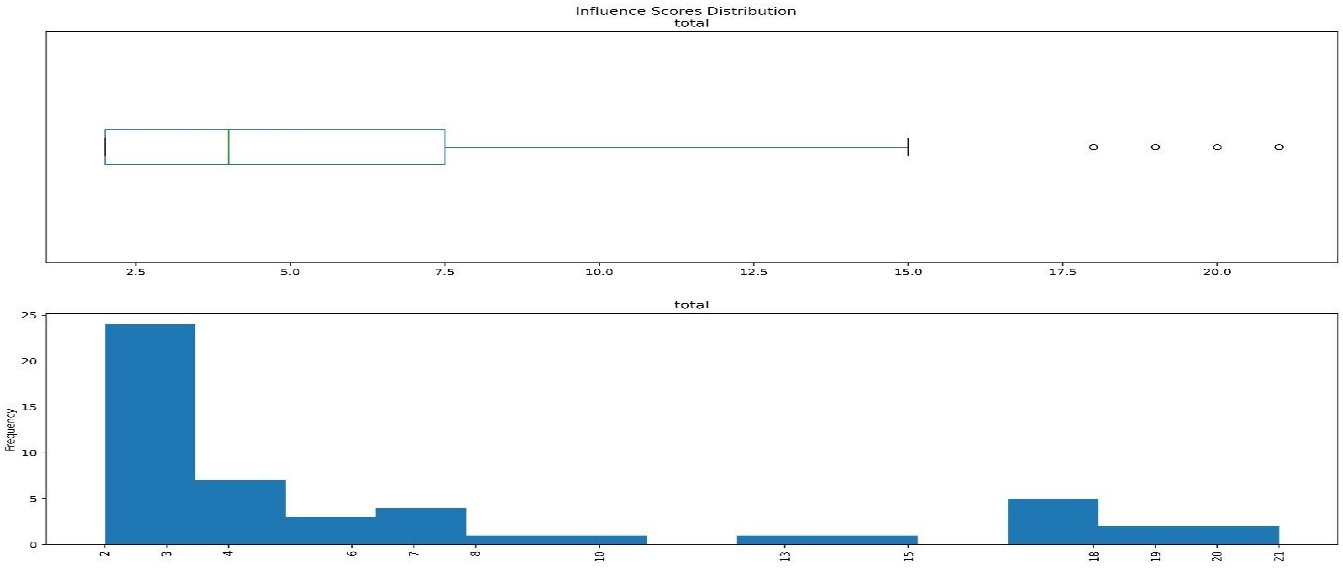

Figure 8, The total score distribution recorded in the edges of the influence graph, constructed on a dataset containing the top 3 newest submissions posted inside the top 3 most popular subreddits on the 12th of July 2021 between 12 to 12:30 O'clock.

Based on the previous testing results, we can conclude that each scoring technique is mostly dependent on the nature of raw data generated by users on a social media platform who 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 a combined scoring technique which can have either an advanced formula or be as simple as an addition of the single scores in each influence edge of the user 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 such as politics, sports 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 classification machine learning model into the process of building the activity and influence graphs.

Due to the normally large number of influences detected between people on online social media, a manual classification between different fields of influence is hard to perform and proof 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 influences into multiple topics learned from a training dataset.

This training dataset can be static over time or continuously changing, the advantage that can be gained be having a dynamic training dataset is its ability to stay updated with the newest details that differs different fields of influence, while a static dataset will be frozen in time and might give false classification results in future use. For this reason, a dynamic process of updating the training dataset before being fed into a text classification model is implemented in this research. More details are included in some of the upcoming sections about the process of crawling dynamic training datasets and building the text classifier used to determine the field of influence based on texts from submissions and comments.

We hereby assume that we have a trained text classifier ready to be used to classify influence edges in the activity graph in between different fields of influence. Then when transforming the activity thread tree 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 tree, along with the texts from the source parent and target child activities of an edge, then input this constructed text into the text classifier to have an estimate about which influence field a particular influence edge 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 as the submission text tends to have a clear indication to the topic of discussion in its activity thread. However, using only the submission text as an input for classification of each activity edge leads to a sterile and little informative classification as all edges between activities in the activity thread will have the same classification, therefore the source parent and target child activities are also added to the input of the classification trial, which also helps detect any derailment between topics under the user discussion in the activity thread.

An overview of this process is drawn in figure 11 below, where 4 classifications had been performed on the 4 edges in the activity thread tree shown on the right of this figure, then labeling the corresponding 4 edges with the respective classification output results in the activity graph at the middle of this figure.

An example of such classification process is the edge between the comment “SC_2” and “SC_3” 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 activity: “But if invested in the right way, it should pay off”
- Child activity: “I fully agree”.

After adding spaces between these 3 texts, the classification method of our trained text classifier is used, which classify the influence edge into the sport category, and then store this output value in its respective edge between users in the activity graph at the middle of figure 11.

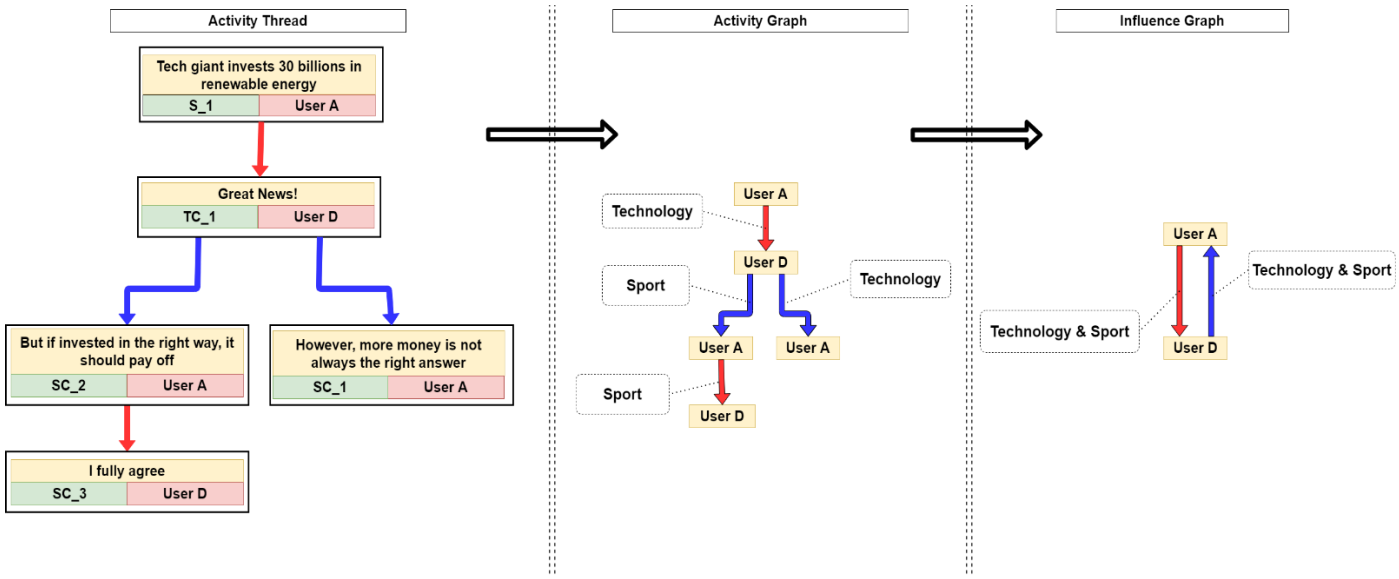

Figure 9, An example overview showing the process of influence field detection integrated in the skeleton of building an activity graph and its final transformation to an influence graph.

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 having the same combination of source and target persons would be merged, and in 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, we might want to manually conclude that the influence constructed from this thread is to be reasonably related to technology, and although the used text classifier has classified some influences in the sport category, 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 discussion under the activity thread, which represents a strength that enable this method to look beyond what the human eye can assume about a certain written discussion.

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, which can have the effect of reducing the runtime of a practical algorithm that implements these three processes all at once, so the need for multiple iterations over records in the dataset is eliminated. However, this can have a moderate increase on the complexity of such implementation, but this complexity can be dealt using best practices under 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 from the website of the "scikit-learn.org" developer community [9], which is a highly optimized model that combines the use of multiple machine learning algorithms for the purpose of text classification. This classifier has a pipeline architecture that starts with text vectorization which maps each unique n-gram[9] 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 text classifier that is to be built on top of vectorization and transformation to produce a text classifier that can be used to categorize given text into different topics after learning 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 implementations of text data analysis in this research was primary guided from Part III of the book titled

[9] An n-gram is a contiguous sequence of n items from a given sample of text. [10]

"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 allow to predict or classify the topic of a new unclassified text. And as most traditional machine learning 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. In the following steps, we will describe the process of gathering training data, testing its initial quality, and tuning the text classifier for optimal classification.

- Gathering training data

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

The top 100 newest submissions on these subreddits are crawled from "Reddit.com" if available, meaning that we can have up to 100 texts from each subreddit, and every group of 3 subreddits can be trusted as a text provider for a certain topic, labeling text from these subreddits with the target topic name as shown in table 1 below.

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

Shared Topic	Subreddits	Number of training records (Up to)	Example record
Politic	politics	100	title: "elections is postponed due to security reasons", label: "Politic"
	PoliticsPeopleTwitter	100	
	elections	100	
Economy	Economics	100	title: "The price of oil is at record low", label: "economy"
	economy	100	
	business	100	
Sport	sports	100	title: "3 days until kickoff of FIFA world cup", label: "sport"
	olympics	100	
	worldcup	100	
Entertainment	movies	100	title: "show to be canceled due to bad weather conditions", label: "comedy"
	comedy	100	
	culture	100	
Technology	technology	100	title: "new material used in batteries might revolutionize electric cars", label: "Technology"
	science	100	
	Futurology	100	
	Total	Up to 1500 records if available	

Following this crawling plan, we obtain up to 1500 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, where training datasets can be obtained on periodic intervals which provide the text classification model with new text submissions that reveals the latest events happening in the different pre-defined categories, making the text classifier always updated with the latest categorization of topics according to the active community and not in fixed a dataset. However, the downsides of this method are the need to constantly consume system and network resources to crawl new training datasets, and in addition having to tune and evaluate the performance of the new 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 tunning of the text classifier.

Before starting to evaluate and tune our text classifier, we shuffle the data randomly to guarantee a fair distribution of categories throughout the dataset and freeze it by using a constant seed in a pseudo-random function, this freezing allows us to have the same results after each run of evaluation, tunning 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.

- Initial Evaluation of the text classification model

We start by testing the initial performance capability that can be obtained with using the crawled training dataset to build a non-tuned text classifier using its default key parameters. The training data is then partitioned into 5 equally sized chunks of data.

A text classifier is then trained and tested N-times where N is equal to the number of data chunks, and 80% or 4 chunks of data are used for training the classifier, while 20% or 1 chunk of data is used for testing the performance of the text classifier. The chunk of dataset that is meant for testing will be different every time a new model is trained, this guarantees that each record in the dataset will participate at least once in the testing process and 4 times in the training process, which increases the representation of every individual record in the dataset, then the we calculated the mean values of measured accuracies, precisions, recalls and F1 scores in the N=5 times the text classifier has been trained and tested, these results are presented in table 2 below and based on a dataset crawl performed on the 13th of July 2021 between 12 to 12:30 O'clock according to the crawling plan in table 1

Table 2, Initial performance evaluation of the text classifier used for influence field detection.

Metric	Measure Value
Accuracy	0.663
Precision	0.67
Recall	0.663
F1-score	0.663

- Tuning the text classification model

When examining the results of the previous initial evaluation of the text classifier, we notice very close values of accuracy, Precision and F1-score parameters at about 70% efficiency which can be improved by tunning key parameters of the pipeline algorithms used in this text classifier, these parameters are stated in the table below along with their significance to the text classifier.

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

Model Parameter	Test 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-2 or 1e-3	A penalty parameter used in the SGD classifier

To find the best parameters for building the text classifier, we run an automated test of cross validation using a split relation of 80% to 20% between training records and test records when forming the 5 data chucks at each possible combination of the test values of tunning parameters in table 3. The result of tuning would then be obtained as a combination of parameter values that gives the highest possible performance in term of the more reliable F1-score. The key parameters that give the highest F-score are those who are most likely to optimize the performance of the text classifier, these optimal values are viewed in table 4 below.

Table 4, Best values of text classification key parameters that gives an optimal and reliable text classification.

Metric	Measure Value
Best score	0.691
vect__ngram_range	0.001
tfidf__use_idf	True
clf__alpha	(1, 1)

Using these optimal parameters, we manage to increase the F1-score score to 0.715 from 0.663, and although this score might be considered low 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 and use for training a text classifier.

Another important question to ask here, what is a good score under testing in artificial intelligence? This depends on what we are trying to achieve from this text classifier, and as 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 to proceed into using this text classifier in building the influence graph and perform an evaluation on its ability to distinguish between different categories of topics in the user's data. 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.

- Detailed information about the performance of the optimized text classification model.

After tuning the text classifier, we repeat the first evaluation step but with using the best score parameters obtained from the tunning process while keeping track of the actual and predicated category of every testing record which is used to produce an informative classification report shown in table 5 below.

Table 5, The classification report after tuning of the text classifier.

Category	Precision	Recall	F1-score	Support
Sport	0.878	0.865	0.872	67
Economy	0.745	0.65	0.694	63
Politic	0.77	0.649	0.704	57
Technology	0.583	0.688	0.631	61
Entertainment	0.61	0.692	0.648	52
Weighted Average	0.723	0.713	0.715	300
Macro Average	0.717	0.709	0.71	300

The classification report indicates a weighted F1-score average of 0.715 with the highest F1-score for the sport category at 0.872 and lowest score to the technology category at 0.631, the lowest score category might has been affected by the big interference of multiple aspects of the other categories into technology, which can cause the model to falsely classify records that should belong to the technology category and not to other overlapping categories.

The confusion Matrix in table 6 below shows the detailed classification of the 300 testing records where 215 records were classified correctly as expected, while 85 records were classified in the wrong category, 51 records or 60% of those with an incorrect classification stands between the 2 categories of Technology and entertainment on the one side against politic and economy on the other side, leaving 34 incorrect classifications between the others more distinct categories, i.e. sport, economy and politic. This confirms that both the technology and entertainment category are often falsely classified to belong to either politic or economy respectively and vice versa, which tells us that this text classifier perform better when classifying general categories with little in common rather than classifying categories that might overlap.

Table 6, The confusion matrix after tuning the text classifier, green colored cells show the correctly classified test records, red colored cells shows the falsely classified test records, cells marked with dark red color shows the distribution of falsely classified test records due to the overlapping between technology and entertainment on the one hand, and economy and politic on the other hand due the natural overlapping of these categories.

Predicted (below) / Actual (Right)	Sport	Economy	Politic	Technology	Entertainment
Sport	58	0	0	4	5
Economy	0	42	0	12	9
Politic	3	5	37	4	8
Technology	1	8	6	42	4
Entertainment	3	1	3	9	36

The objective of analyzing social influence aims to find the most non-overlapping category of influence in society like politic, economy and sport. Each one of these categories are clearly defined on what they are about, this makes our text classifier suitable for use in building a social influence model with good potential of detecting the right category of influence.

In the next section, we will evaluate the performance of this text classifier under the process of forming the influence graph using real life data crawled from "Reddit.com" that is new and unseen to the text classifier. As we aim to investigate how well this classifier is contributing to the objective of identifying different fields of influence 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 social media into distinct 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.

We start by crawling the 3 most popular subreddits on reddit at the time of this evaluation which are "Home", "AskReddit" and "PublicFreakout", then extract information about the 3 newest submissions and their comment threads, along with a training dataset for the text classifier which includes 1500 records distributed between topics according to the crawling plan detailed in table 1. All of the information attributes about the subreddits, submissions and comments are stored in the data structure visualized in the Entity-Relationship model established in figure 2.

After having a real-life dataset to work on, an activity thread is constructed, then transformed to an activity graph that is used as a foundation to detect influences between distinct persons showing both the strength and predicted field of influence.

In this evaluation, we examine the different predicted fields of influence between the participating persons by focusing on the influence edges regardless of which persons 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.

Before preceding into studying the edges of the influence graph, we are interested of knowing how many submissions were crawled from each subreddit or group, which is important to make sure that no one group is overrepresented in the dataset, we will call this distribution measure for "Crawled Groups". And in addition, we also wish to know how many influence edges are learned from each group, and we call this distribution measure for "Modelled Groups". Finally, we extract all predicted fields of influence from the edges of the user influence graph and visualize its distribution, this distribution measure is called "Predicted Influence areas".

The measure of "Modelled Groups" can help us understand the observed distribution of the "Predicted Influence areas", for example if we have a subreddit or group that have a reputation to be related to a certain topic like for instance sport, then we would expect around the same percentage on this topic in the distribution of both "Modelled Groups and "Predicted Influence areas" assuming all of the crawled and modelled groups are related to different categories.

Now that we have defined our evaluation plan to test the performance of influence field detection in the influence graph, we visualize this information through three pie plots, one for each distribution measure as shown in the next figure

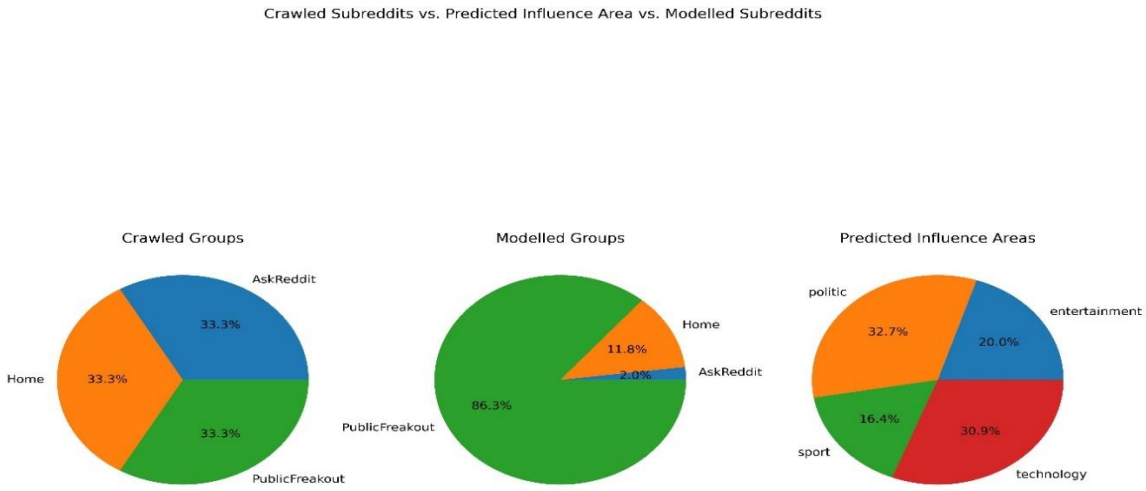

Figure 10, A visualization of the distribution of targeted subreddits (left), the amount of detected influence in each subreddit (middle), and the distribution of detected influence fields (right) in the influence graph. The foundation dataset of the influence graph is based on the 3 newest submissions on the 3 most popular subreddits at the time of this evaluation.

The right plot in figure 12 above proofs that our text classifier has been successfully able to identify 4 out 5 possible topics of influence between persons in the influence graph. The absent topic is economy which may be expected since all the 3 most popular crawled subreddits are not strongly associated with economy but more with the 4 other topics which are technology, politic, sport and entertainment, with relatively higher portion of politic and technology.

The results above are promising for the reliability of our text classifier to be used in building the user influence graph. However, we wish to be sure that our model can distinguish between all the target topics in a training dataset of the text classifier. For this reason, we will crawl the 3 newest submissions from 5 other selected subreddits where each subreddit is strongly associated with one distinct topic of influence, and strongly not associated to the other 4 topics, this means we will be crawling the subreddits; “Finance”, “Cinema”, “worldnews”, “research”, and “NBA” as each of these subreddits is strongly associated with one of the topics; economy, entertainment, politic, technology, and sport respectively. This should enable us to see whether our model can detect all the 5 different categories.

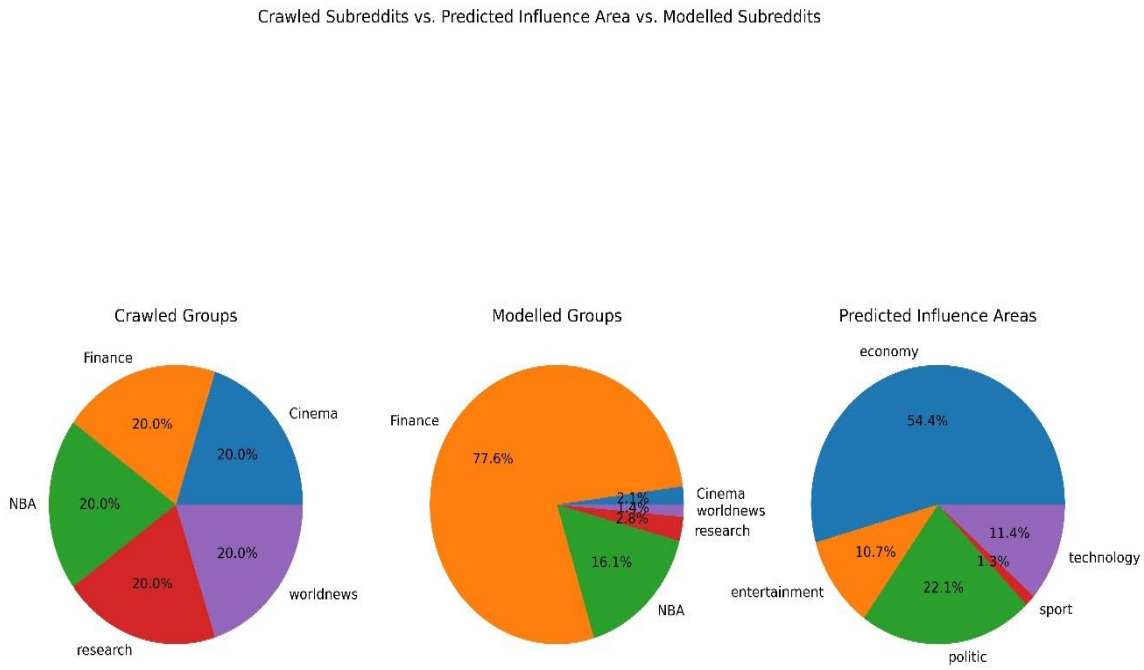

Figure 11, A visualization of the distribution of targeted subreddits (left), the amount of detected influence in each subreddit (middle), and the distribution of detected influence fields (right) in the influence graph. The foundation dataset of the influence graph is based on the 3 newest submissions on 5 selected subreddits that is most likely to be assiciated with one of the targeted influence fields in the training dataset at the time of this evaluation.

As we see from the plot above, 3 submissions were crawled from each of the 5 subreddits, and after building the influence graph, all influence edges were reviewed to know how many influence edges each subreddit has contributed in, then the same process was performed to know the share of each predicated topic between all the influence edges.

This shows that the text classifier has helped identify the 5 different categories with greater share for economy which is expected since Finance is a subreddit that is mainly associated with economy. However, there is only 1.3 % influence edges that is predicted to be in the field of sport, but NBA does have a greater representation in the influence graph than cinema, worldnews and research. Knowing the exact reason for this observation requires digging down into the nature and origin og each individual influence edge. However, some difference and variations between a certain subreddit and the topics of influence extracted from it is not necessarily a downside since most topics tends to overlap in real life scenarios.

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 his connections to other people in the 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 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 neighborhood to examine the flow of possible paths throughout the entire graph.

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

There exist a wide range of available algorithms used to calculate centrality measures, some of these focuses 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, and following with an implementation of the betweenness centrality which looks for the occurrence of each user node in the available 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 an 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 more simple 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 from the same node. And although both 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 have 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, regardless of how many other persons does have an influence on an 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 14 below shows an example of an influence graph, where degree centrality measures are calculated for each user. We notice that the top influential users 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 each of them has, "User E" on the other hand is only a receiver of influence and does not influence any other persons which gives him the lowest influence score of 0.

- Betweenness Centrality

Betweenness centrality is more concerned of the participating role of each person node in all the possible shortest paths between any given two users in the influence graph. Following this principle allow us to rank each person based on his ability to carry 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, not to include the nodes of the source and target persons in occurrence counting, each person will then have his rank to be equal to the number of times it occurs in connecting nodes of any shortest path in the 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 the figure 14 below, and it elects "User F" 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", and in the second place "User B" and "User C" gets the score of 3, while the rest of 4 non-central users

gets the lowest influence rank of 0 because they do not participate in connecting other persons in the influence graph.

- HITS centrality – Auth and Hub with 10 iterations

The process of the HITS[10] 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 give 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, then to calculate the new auth scores each node gets the sum values of old auth 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, and for the purpose of normalization the given new values of auth scores are summed up and each new auth score is divided by the sum of auth scores to output a normalized auth score for each node, the same process is repeated on the new hub scores but by summing up the values of the new hub scores. In the next iteration, the new auth and hub scores will be marked as old scores and the same two processes of auth and hub score calculations is repeated.

In theory the HITS algorithm can carry on for an infinite number of iterations, but in practice both values of auth and hub scores would converge each to its approximate value, then the calculations can be stopped, 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 auth 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 both journals and web pages which can point to each other by using references in their text content. The nature of our user influence graph is very similar to the usual application of the HITS algorithm, as both the nodes and links of our developed influence graph are homogeneous, i.e., nodes does only represent different persons, while directed links between them only represent social influence 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.

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

Both graphs in the bottom of figure 14 below shows the results of running the HITS algorithm with a limit of 10 iterations to assign each node a hub and auth score. On the right side, both nodes of "User B" and "User C" were given the highest hub score indicating their ability to point to other nodes in graph, while examining the auth scores in the graph on the left side reveals that "User F" is the top scored in the graph, this is highly expected as a good authority in the influence graph is a person that is linked by many different strong Hubs which are "User B" and "User C" in the right graph at the bottom showing the hub centrality scores.

[10] Hyperlink-induced Topic Search.

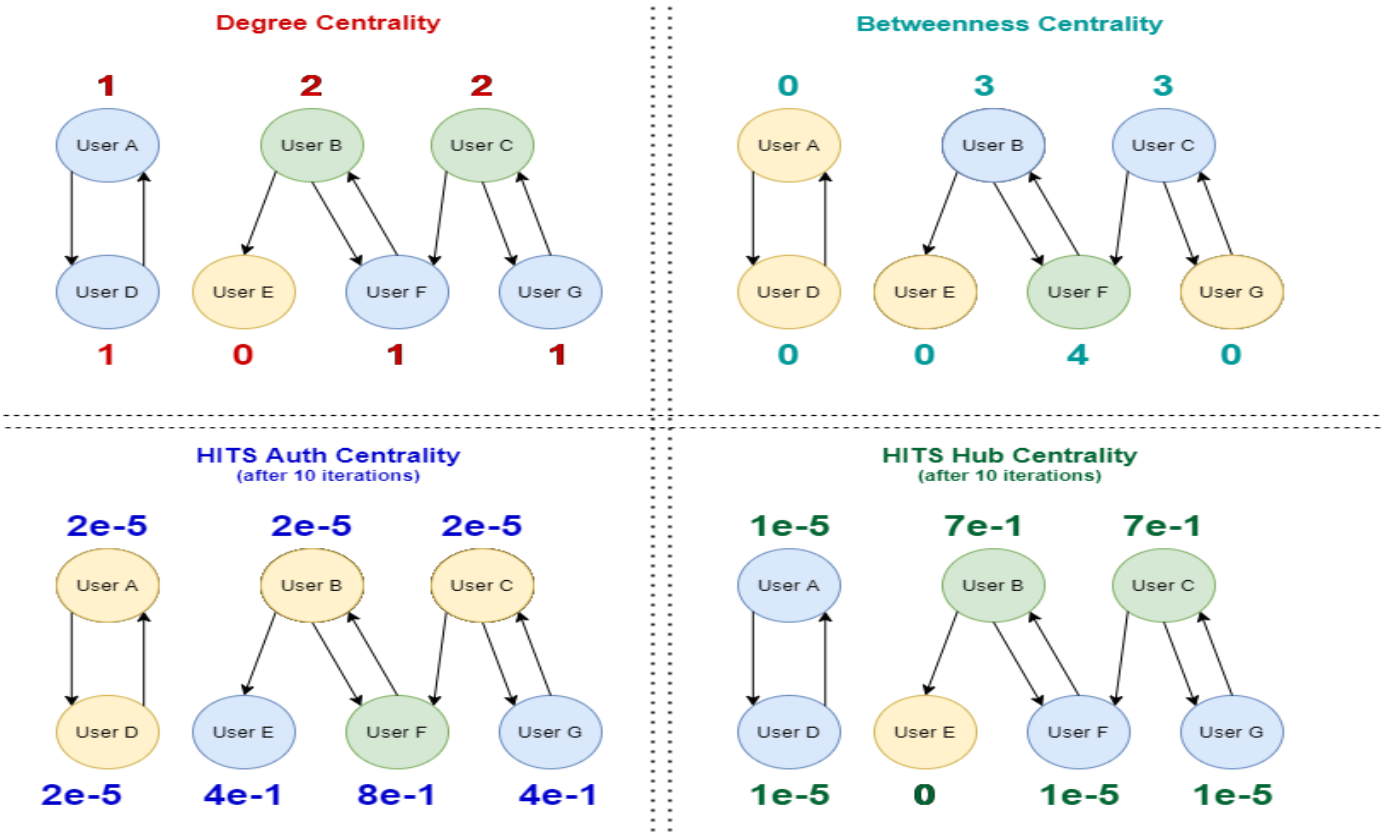

Figure 12, An ranking example using 4 unweighted and direct centrality measures; outdegree centrality (upper left corner), betweenness centrality (upper right corner), HITS Auth (lower left corner), and HITS Hub (lower right corner).

We hereby conclude that the auth scores helps detecting 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 hits scores to the results gained from the degree centrality on the same influence graph, we notice that both techniques did yield the same ranking for every person in the graph, which confirms that the hub centrality is a measure of how influential a certain person is in the influence graph.

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

At this point of insight, we notice that each centrality measure has a series of features when used for detecting the most influential users 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 the user influence graph to understand the capability of each centrality measure that enables this analyzer to make the best decision on which person is to be considered the most influential. The table below summarizes those previously discovered and discussed features of the various centrality measures.

Table 7, A comparison between 3 types of centrality; outdegree, betweenness, HITS Auth and HITS Hub. This comparison is based on the observations of ranking performed on the example influence graph in figure 14.

	Degree	Betweenness	HITS Hub	HITS Auth
Reveals	Direct Influencers	Influence transformers and connectors	Direct Influencers	Influence transformers and connectors
Favors	Number of outgoing influence edges	Users at the middle of a connected segment of the graph	Number of outgoing influence edges	Users at the ends of a connected segment of the graph
Simplicity	Easy	Medium	High	High
Interpretation of results	Easy	Medium	Difficult	Difficult
Running time	Low, as it only requires one iteration over the edges in the graph.	Medium, as graph traversal is needed to find the shortest path between every two nodes in the graph	High, as it requires multiple iterations to achieve convergence	High, as 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 on which persons to be considered the most influential among others in the influence graph.

Before moving forward in analysis, it is important to highlight the similarities and differences between using the scores of outgoing influence edges from one node and using its centrality rank as a proof for its influence. Centrality measures are calculated on a larger scale than the scores on influence edges, as centrality measures either examine a group of connected nodes like a 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 like 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, and not to forget that in order to include a certain person in the influence graph, at least one interaction between persons has to be registered.

This leaves us to wonder what to use in order to find a good influencer, the best answer is to combine the use of some or all of the available measurements to establish a decision, without forgetting to evaluate the objective of the analysis itself, i.e., if we wants to find the top influential users that helps spread influence throughout the network even though they are not necessary the source of this influence, then the betweenness and HITS Auth centrality 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, than 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 to enable analyzers to easily point out the area where the highest scores of influences occur. And then look at the centrality ranks to make a stronger decision about which person are to be elected as the most influential.

An example scenario is given in figure 15 below based on crawling the top 3 newest submissions of the selected subreddits (worldnews, Finance, NBA, Cinema and research) where influence edges of high total score (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.

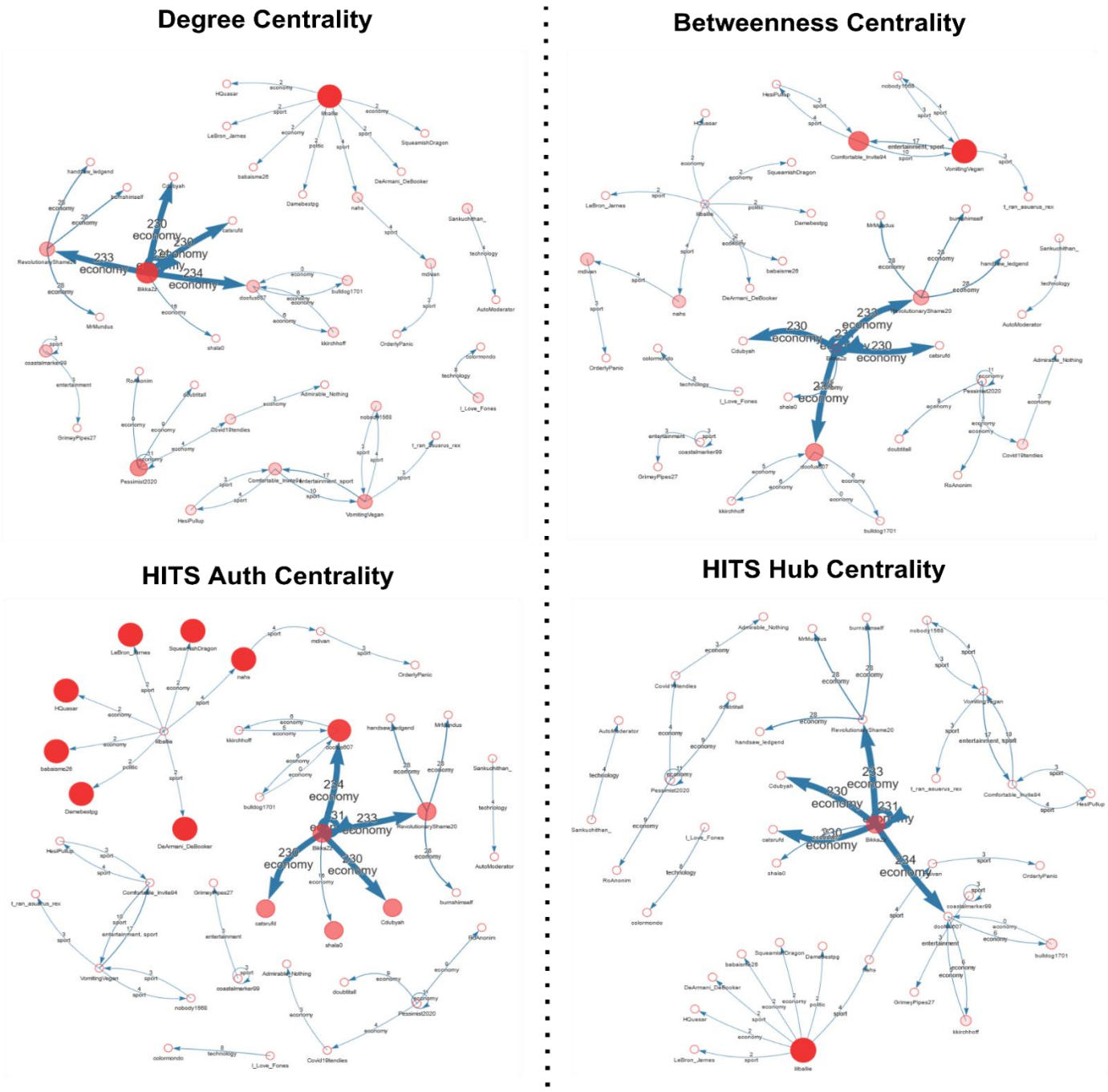

Figure 13, A visualization of an influence graph to compare between the use of node size and transparency to indicate the ranking of outdegree, betweenness and HITS centralities, against the use of edge thickness to indicate the total influence score that is a more direct and personal influence between persons in the graph.

Using the degree centrality to rank persons in the upper left graph of figure 15, we notice that the top ranked person does not have any strong influence compared to the second top ranked person, this is because we are using the unweighted version of outdegree centrality because we wish to relax the weight of 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 nodes and others with lower centrality values.

On the other side and examining at the upper right graph in figure 15, which visualizes the ranking of betweenness centrality, we notice as expected that it has a tendency to give a higher centrality scores to those persons who are able to connect other remote persons in a shortest path, this is however not the

case for the graph using the HITS Auth ranking at the lower left side of figure 15, as persons at the end of an influence path tends to gain a stronger rank, which is good if we are trying to find the most influenced persons in the influence graph.

Table 8 below shows the rank of the top 3 most important persons of the 4 graphs in figure 15 above. Where both measures of the outdegree and HITS-Hub centralities had ranked the same two persons on top marked with the green colored cells, however they differ with ranking different persons in the third place, which is a confirming results because we are expecting both 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 as in the third place where different persons are ranked when changing from outdegree ranking to HITS-Hub ranking.

When comparing the betweenness rankings to the HITS-Auth rankings on the same graph in table 8, we notice a single common selection of one person in the second place using HITS Auth, and in the next third place using the betweenness measure, marked with the red colored cells. Other from that, there is a great variety in selected users which is also a desired result to reveal as much information as possible using different measures without having a total differ in their outcome.

It is also interesting to notice that one person has been ranked in the second place using both the outdegree and HITS-Hub measures which shows that he is a good direct influencer, and the same person was also ranked in a group of third place persons using the HITS-Auth measure which tell us that this person is good in connecting hubs together and perhaps transferring influence across the graph despite not being selected from the top users using the betweenness algorithm. This person is marked in the blue colored cells of table 8.

Table 8, The ordered rankings of the top 3 most important persons in the influence graph in figure 15 using the 4 centrality measures; outdegree, HITS Hub, Betweenness and HITS Auth. Colored cells are used to mark the same persons in the ranking across these 4 centrality measures.

Rank	Outdegree	HITS Hub	Betweenness	HITS Auth
1	**Lilballie**	**Lilballie**	VomitingVegan	Nahs
				SqueamishDragon
				LeBron_Jarnes
				HQuasar
				babaisme26
				Damebestpg
				DeArmani_DeBooker
2	**BikkaZz**	**BikkaZz**	Comfortable_Invite94	**doofus607**
3	Pessimist2020	bulldog1701	**doofus607**	**BikkaZz**
		Kkirchhoff		Cdubyah
				shala0
				Catsrufd
				RevolutionaryShame20

Now that we have went through a practical example of analysis and investigated the common and uncommon features between edge scores and centrality measures. We shall say that to correctly rank the most influential person in the influence graph, it is crucial to understand the ability and features of the used centrality measure and their impact on the final ranking results, and that those centrality measures should be studied hand in hand with the individual influence edges and their selected score, and perhaps influence field. This might sounds to be a complicated process of analysis but when using the right visualization tools the display of the user influence graph can become more intuitive and easy to understand as we have seen earlier in figure 15.

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 in visualizing the picture of influence flow in the studied dataset.

We start by drawing the activity graph on 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.com statistics by using the python Reddit API wrapper on the 12th of July 2021 between 12 to 12:30 O'clock.

The activity graph has 3 types of nodes: the submissions, top level comments and sub level comments. In figure 8 below, we view the activity graph of this Reddit dataset and notice a little higher concentration of top-level comments having the red color in comparison to sub level comments with the blue color, which tells us that people in this dataset have a medium engagement in the threads of submissions, this engagement benefits the process of detecting social influence in the dataset when constructing the final person-oritened influence graph.

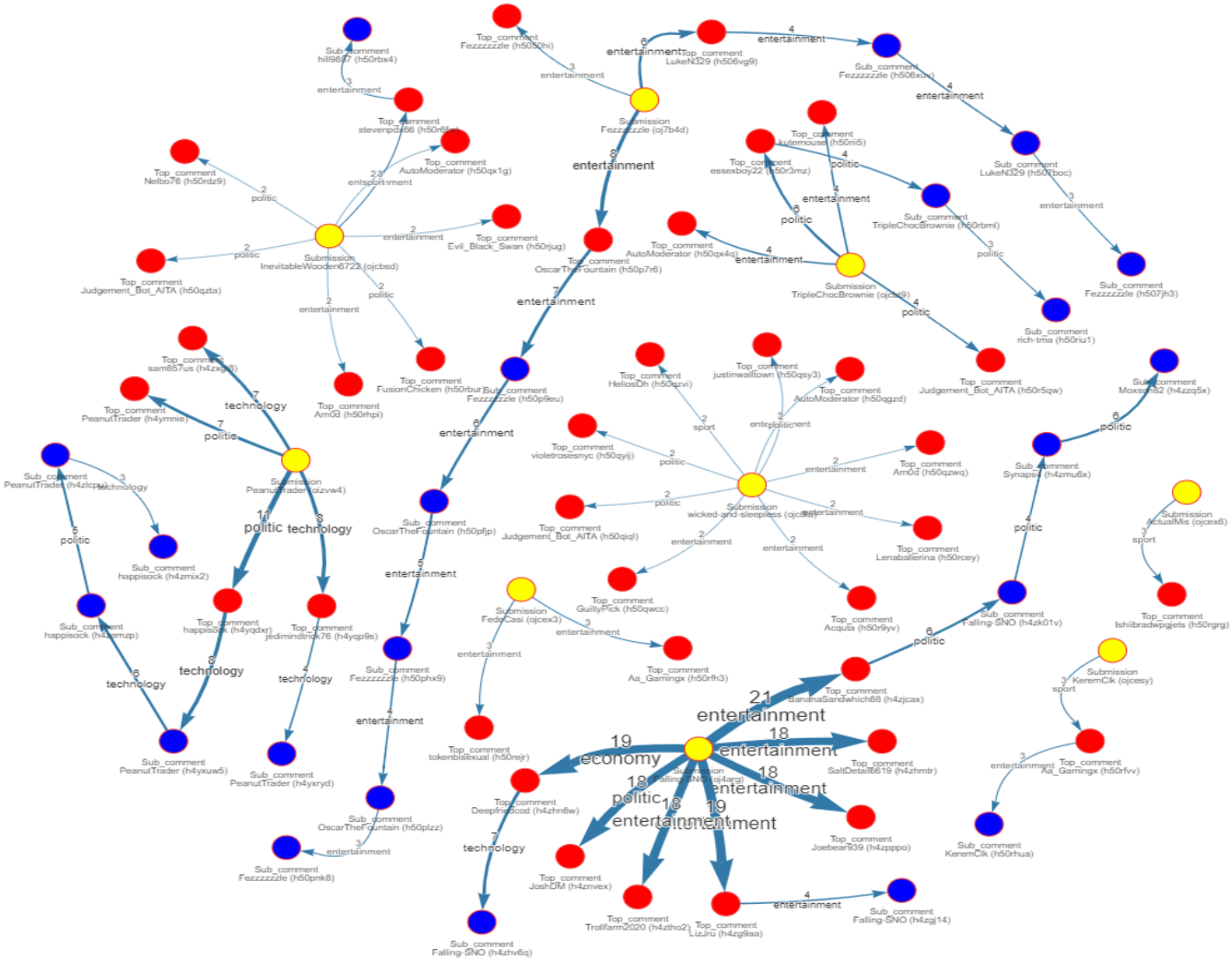

Figure 14, The activity graph, having the colors; yellow for submissions, red for top level comments, and blue for sub level comments. The graph is constructed on a dataset containing the top 3 newest submissions posted inside the top 3 most popular subreddits on the 12th of July 2021 between 12 to 12:30 O'clock.

The nodes of the activity graph in Figure 9 represents 69 activities in the crawled dataset from Reddit, having the count of 9 submissions, 38 top level comments and 22 sub comments. Since the percentage of sub comments is around 32%, it can be used as an indication of good social engagement from the participating persons in the dataset. This result is also highly visible by taking a quick look at the colored activity graph in figure 9 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 influence edges to transform it to a person-oriented graph, we obtain the influence graph shown in figure 10 below. The visualization of this graph takes advantage of the total combined score values to adjust the thickness of each influence edge, indicating its high or low score value in an intuitive display that allows analyzers to identify and focus on the high score influence.

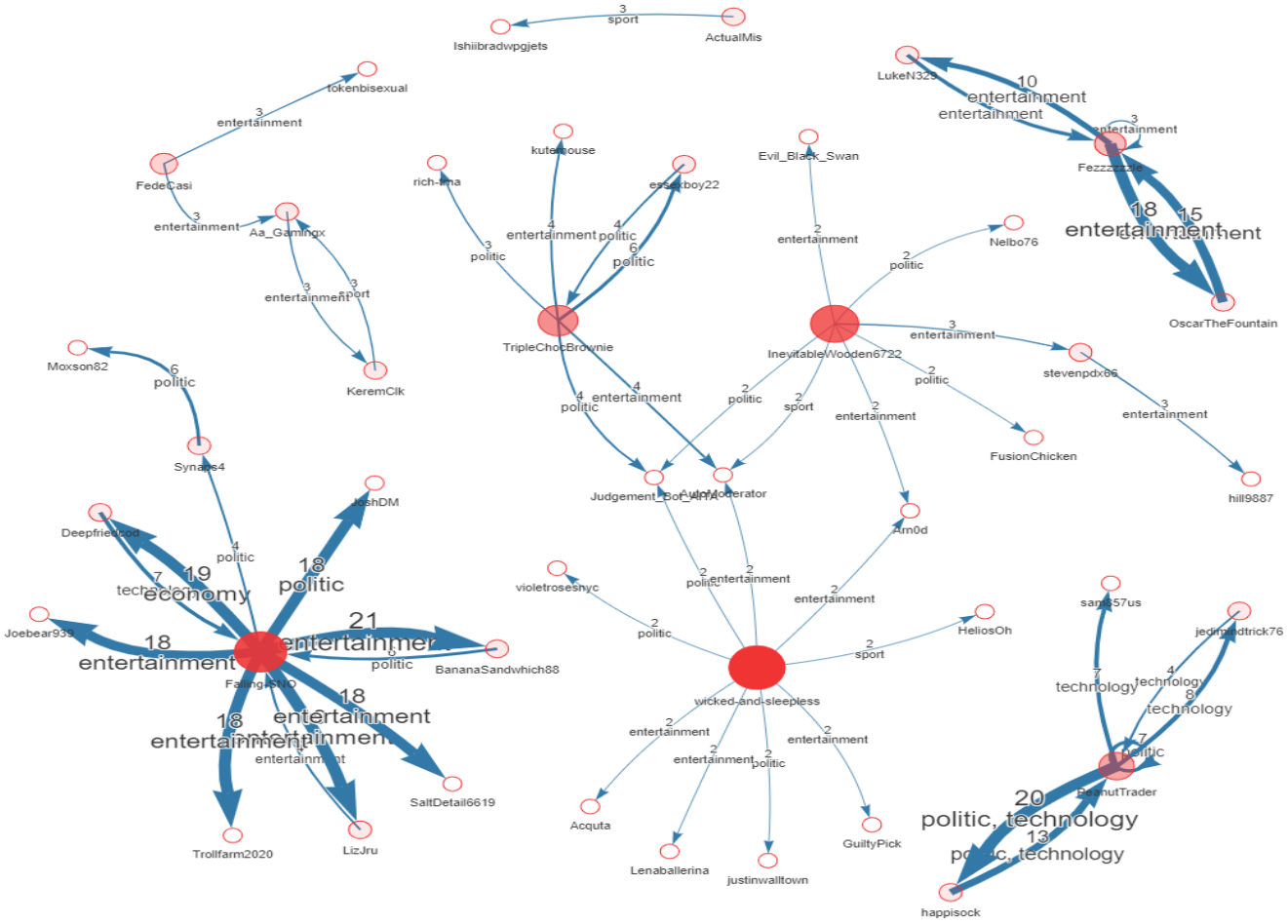

Figure 15, The influence graph. The graph is constructed on a dataset containing the top 3 newest submissions posted inside the top 3 most popular subreddits on the 12th of July 2021 between 12 to 12:30 O'clock

In addition to visualizing the strength of each influence, the type on influence is also used as a label on each influence, 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 will accumulate multiple influence types as viewed in the right bottom of the influence graph in figure 10, this tells us that this influence is most likely to be in politics and 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 opacity of the person node as in figure 10.

The use of unweighted centrality measures allows the analyzer to have a view that is dependent from the score of influence edges, as we observe in the influence graph in figure 10, where 3 significantly bigger person nodes are shown in the middle of the graph where no thick edges is to be found. But in the case of having the combination of a series of thick edges connected by a big sized person node in the graph, then it is a good indication of the high influence this person has on other connected persons. An example of this case is to be observed at the left bottom area of the graph on figure 10 above.

Looking at the overall collection of different features and tools provided by the final influence graph, it is clear to observe its beneficial capabilities under the process of social influence detection and analysis. Many applications can then benefit from easiness of interpreting the influence graph to determine the strength of influence, its predicted field of influence along with the centrality ranking of persons in the dataset.

V - K. Future Improvments of The Influence Graph Model

Online social media is developing rapidly, and it is therefore 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 between users of 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 functionality, as it is also possible to further develop the modelling 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 an 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 an 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 us to edit 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 is able to 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 algorithms in sentimental analysis.

Furthermore, it is also possible to consider taking in 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 single classified field of influence.

Drilling down into our already implemented text classifier, we can also look for more improvement for accuracy and F1-score upon tunning and evaluation by adjusting the tunning range of model 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 difference in training datasets can lead us into falsely choosing the least effective text classifier.

For ranking persons 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 persons in the model of influence graph, and 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 tunning, testing and change management should help achieving this objective and keeping the produced model of the influence graph both relevant and informative.

VI. THE TECHNICAL IMPLEMENTATION

VI - A. Design Architecture

A desired outcome of this project is a technical IT system that can identify and classify influencers and their fields of influence on periodic basis, 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 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 easy to access and an intuitive web interface. This technical solution consists of four segments corresponding to the nature of the main process taking place in each segment.

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

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

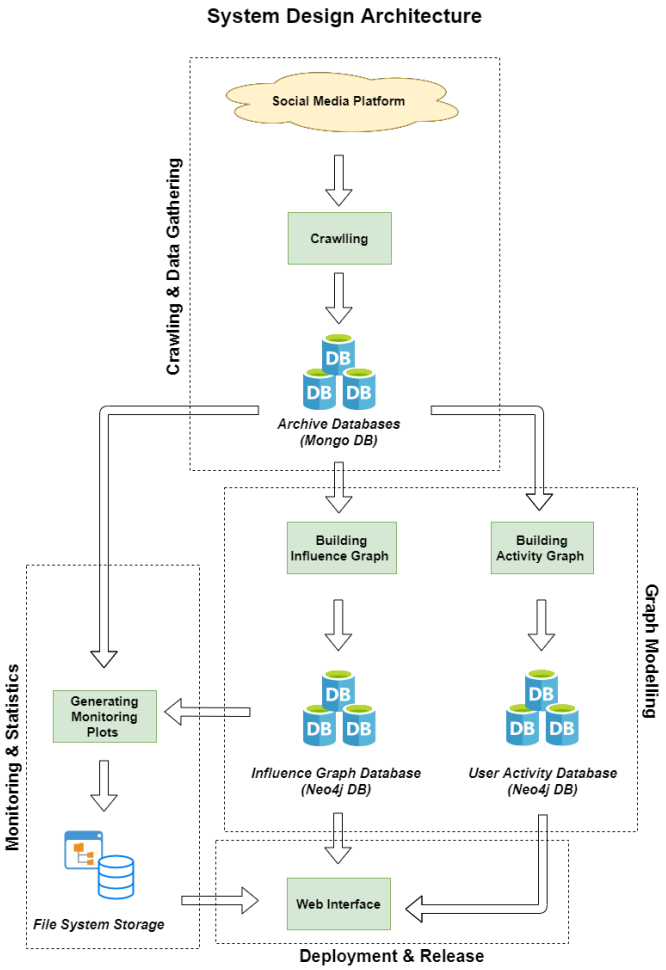

Figure 16, The system design architecture of the technical implementation of this research.

To store crawled data in a more general and modern data structure, we take in use the document NoSQL mongo database which stores information about the groups, submissions and comments in a simple data structure that is very similar to the structure of JavaScript Object Notation, also known as JSON, which has proven capabilities in modern IT systems due to its easiness 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-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, such as filtering the influence graph by a score range or certain types of influence.

This implementation is published under the domain https://smia.uis.no for the purpose of demonstration and is expected to be alive until the 15th of June 2022. This site performs two separate reddit crawls each day; one that crawls the top 3 newest submissions on the top 3 most popular subreddits starting at 12 pm. Oslo time, and the other crawls the to the 3 newest submissions made on five 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, test or production environments of this application.

VI - B. Crawling & Data Gathering

The phase in the technical solution where data is downloaded from a remote social media and stored in the local archive databases is called the crawling phase. It 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, and also makes it easier to 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 given 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 crawling patches with respect to the continuous daily timeline.

Further on, data from each crawling patch 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, and in each of the databases the date of 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 below shows the progress plan of a crawling job

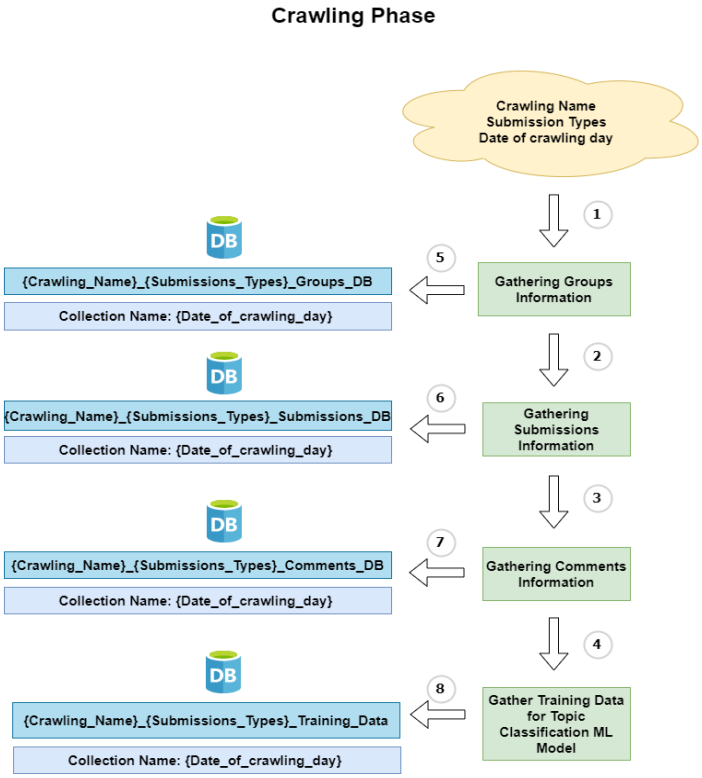

Figure 17, Storage plan of the daily crawled datasets, please note the numbered sequence of crawling and data storage used to avoid any inconsistency in stored datasets.

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 of having an incomplete dataset because of a failure or loss of connection under the transfer of data from a remote social media 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 database, 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 its submissions and comments along with targeted training dataset used for influence field classification. Additional methods can also be implemented as an extension if needed. The crawling classes belongs in the "crawling" module located in the "classes" folder of the source code that forms this technical solution.

An example of such 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 task where crawling, modelling, and other phases are performed 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 from the top 3 most popular subreddits, which helps clarifying how the crawling phase can be applied for future crawling targets and their respective classes.

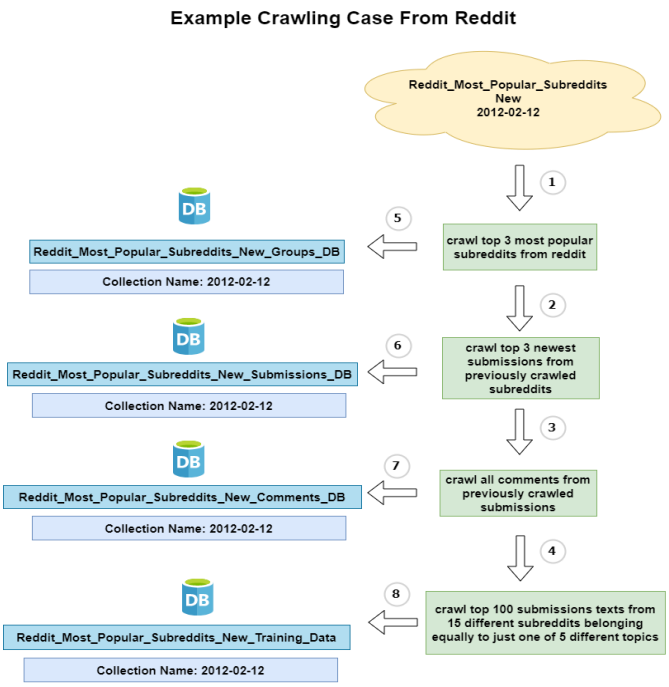

Figure 18, An example of the progress in the crawling phase for crawling the 3 newest submissions on the top 3 most popular subreddits.

VI - C. Implementation of Graph Modelling

The next phase after processing and archiving the crawled data from a social media platform is to apply our modelling algorithms on this patch of data to produce the activity and influence graphs. Both algorithms are explained with detailed examples under the main section of influence graph modelling 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 modelling phase, where in step 1 to 4, data is fetched from the Mongo archive databases using the unique parameter combination of a crawling patch. Then in step 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 inn procedures that can be triggered to activate many algorithms of graph data science such as calculating the centrality of nodes in the registered graph using a wide variety of centrality measures like degree, 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.

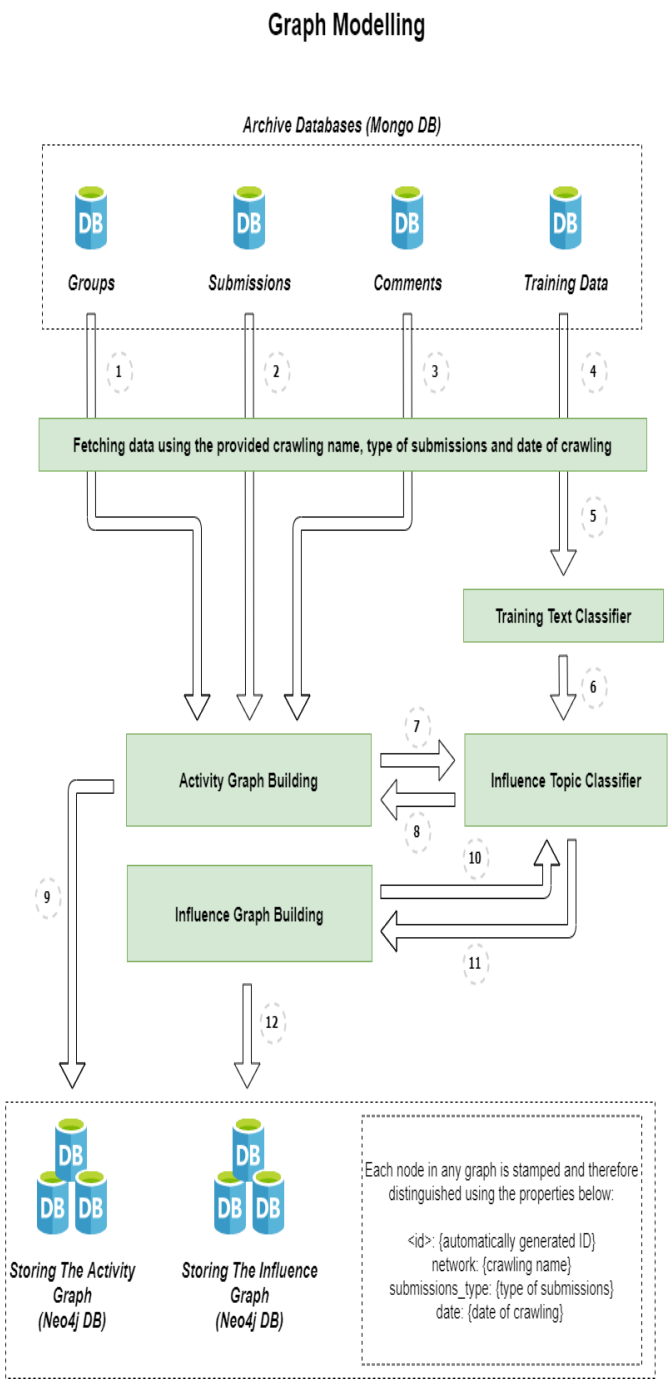

Figure 19, A detailed sequential overview of the different main processes during the periodic graph modelling in this technical implementation.

The source code that implements this graph modelling phase is to be found in the module called “modelling” in the “classes” source code folder. In this module, we find the classes listed in table 9 below with a brief description of its purpose in the technical application.

Table 9, A brief description of the integrated classes in the module specialized for graph modelling in the source code of this technical implementation.

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 determining the field of influence based on the texts of activities as input. In this class, evaluating then 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 report method about the evaluation and tuning processes and a classification method are also included in this class.
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 people 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, reliability, and integrity of revealing the full picture of influence flow between people.

Monitoring and viewing of continuous statistics about both nodes and edges of daily influence graphs is integrated into this technical implementation in the module called "statistics" located in the "classes" folder of 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 edge scores distribution, along with a pie share overview of social media groups and detected type of influence in the graph, these last plots were used earlier in this research as intended in the test and evaluation process in different aspects of the influence graph.

The generation of statistical plots is triggered in the driver script after the process of modelling and 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 shows 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 help explain it by visualizing the distribution of submissions, top level comments and sub level comments in the crawled dataset using different node colors.

VI - E. Deployment & Release To The Web Interface

The deployment and release processes are highly automated to publish the results of daily built graphs and 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) endpoint routes that gives 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 desired range of scores and a desired range of influence fields or types.

Many of these routes 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 to its current and potential users.

Another feature of the web interface gives the analyzers the opportunity to view the evaluation and tunning reports of each text classifier used in influence field detection, which is dedicated to a certain daily crawling patch of some 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.

VI - F. Building a Driver Script

Now that we have segmented the way this application receives, process, and retrieves data, it is time to put all these pieces together to produce an example driver code dedicated to a specialized crawling job like crawling the top 3 most popular subreddits by processing their top 3 newest submissions on periodic basis like for instance once each day.

An example of such crawling job is included at the root directory of the source code inside the script called "reddit_most_popular_subreddits_driver.py", this script can be used as a template example in the recommended way of producing a new crawling job on this application.

The driver code starts with configuring an execution plan which defines the network name and submissions type that distinguish this crawling job from others. In addition, it is obligatory to define the different stages of this script in the "stages" array that 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 exec plan, to perform the following 4 technical steps of this solution:

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

2. Influence Modelling

   This step uses the class for influence graph modelling 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.

3. Activity Modelling

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

4. 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 Step 2 and 3 are both dependent of data stored in the Mongo archive databases during the process of crawling. And further on, step 5 depends on having results from 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 in 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 pre-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 vital for maintaining the health of IT systems. It is therefore important to log any exceptions triggered under the execution of system procedures and operations.

This technical implementation implements error handling on a global execution level at each script of a driver code. Exceptions that might occur here should be caught by using a try-except control structure and then written to the “Logs” directory at the root of the source code. In other words, errors are not stored in dedicated databases but uses 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 errors is called “Logs” as mentioned above.
2. The children of the directory “Logs” can be numerous dated directories representing the actual date a crawling job runs on, an example is “2021-05-20”.
3. Inside any dated directory, multiple directories can be created for each crawling job.
4. Finally, a directory for each types of submissions is created under the directory of network name and the “error.log” file is stored at this path.

Also notice, that it is not always necessary to create the path of inheriting directories needed for storing an “error.log” file, because if a crawling job executes without errors as expected, then there will be no directory path created and neither a error.log file. The creation and storage of error logs is implemented to automatically create the directory path and an error.log file, then store the full thread of the captured exception in it, the python snippet code below demonstrates how logging is performed in this technical solution with the help of 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)
```

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 at the same hosting environment, which results in an easier and straight forward debugging experience.

VI - H. Data Protection

Protecting data is an integral part of most IT systems, where it is important to maintain the integrity and confidentiality of system data to guarantee a high level of availability and service integrity to the 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 possibility to reuse previously crawled data from social media with respect to the date of crawling.

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

Data consistency in its structure and content is vital to guarantee a normal operation of this technical solution, it is therefore 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 where this 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 in the source code. They are imported from the driver scripts for writing and reading access, and in the server script of the web interface for reading access from these database servers of "mongo" and "neo4j" respectively.

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 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 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 system database servers from outside the host machine. And 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 need of the system administrator and the purpose of the application. An example here, is to consider restricting access to the web interface by implementing a proper authentication method like the HTTPS basic, two-factor, or one-time-password authentications. HTTPS basic is implemented in this technical solution because of its simplicity and acceptable level of security when used together with HTTPS/SSL. More details about the implementation of basic auth is given in the upcoming section on Basic Authentication over HTTPS/SSL.

VI - I. Caching

Public clients using the web interface of this technical solution has the need of reading all produced graphs in an efficient and fast way regardless of their size and complexity.

Because of the relatively big datasets this system is dealing with, users can experience high latency when using some routes made available by the web 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" on routes that has the potential of struggling with latency after receiving requests clients, an example route 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 influence classification model.

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

1. CACHE_ON

   A "False" value of this parameter switches the caching mechanism off.

2. CACHE_TIMEOUT

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

3. CACHE_DIR_PATH

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

An open-source python module named "flask-caching is imported 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 in the code snippet below.

```
# 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():
# 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
```

As we see from the previous snippet code, caching is mounted to work on the route "/topic_detection_model" with the timeout configuration in the ".env" file, specified by the system administrator. In addition, the query parameters from the user request are also included in caching so it would be possible to produce a cache 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 route.

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

The entire cache storage of the system can be cleared using the protected route "/clear_cache" or refreshed by crawling patch using another protected route named "/refresh_cache". The reason these two routes are protected is because an attacker can take advantage of continuously triggering the cleaning and refreshing of cache records, which can result in consuming a lot of resources on the host machine and increasing the response latency due to the lack of cache records.

In addition to giving the system administrator a protected route to clean and refresh cache records with, it is probably 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, where an instance of this handler can then be initialized and used to clear the specific cache records for a crawling patch 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 this approach, it is a recommended to set the timeout of cache records to forever by giving the environment variable "CACHE_TIMEOUT" a value of 0. This way cache records would be refreshed when needed after a change has occurred on the model graphs or any other result of the running driver script(s).

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 web interface where many equal requests are sent to the web server from different clients, and 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 parameters. This use of caching has a great value of increasing the availability and reliability of this technical implementation.

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 low, this increases the responsibility on this solution to protect the integrity and confidentiality of 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 relay on the HTTPS/SSL encryption to perform such 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 start of the UI sever and used to configure credentials using 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 inn on 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 users and by 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 https to guarantee both the identity of the site server, and that HTTPS is used for an additional strong 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] is available in the book titled "Cryptography and network security: principles and practice" which has provided enormous help by guiding the process of securing the user interface of this application.

In this published demonstration of this solution, we are only crawling publicly available data from "Reddit.com", and this data is covered with the anonymity that reddit offers its users, because of this, routes to view graphs and many other analysis in the UI is open for public, 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 authentication, or consider implementing "two-factor" or "one-time-password" authentication for more security and access control on the user interface.

VI - K. Supplementary Tools & Features

The source code of this technical solution is supported by 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.

- 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 the 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 class.

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", and it has a timestamp, and it is marked with the unique key parameters of the crawling job specified in the driver code.

This register is also used by the statistics module to retrieve information about how much time it took to crawl the groups, submissions, comments, and training data for text classification. The figure below shows an example plot of crawling runtime measurements which is accessible to public users of this system via the web interface.

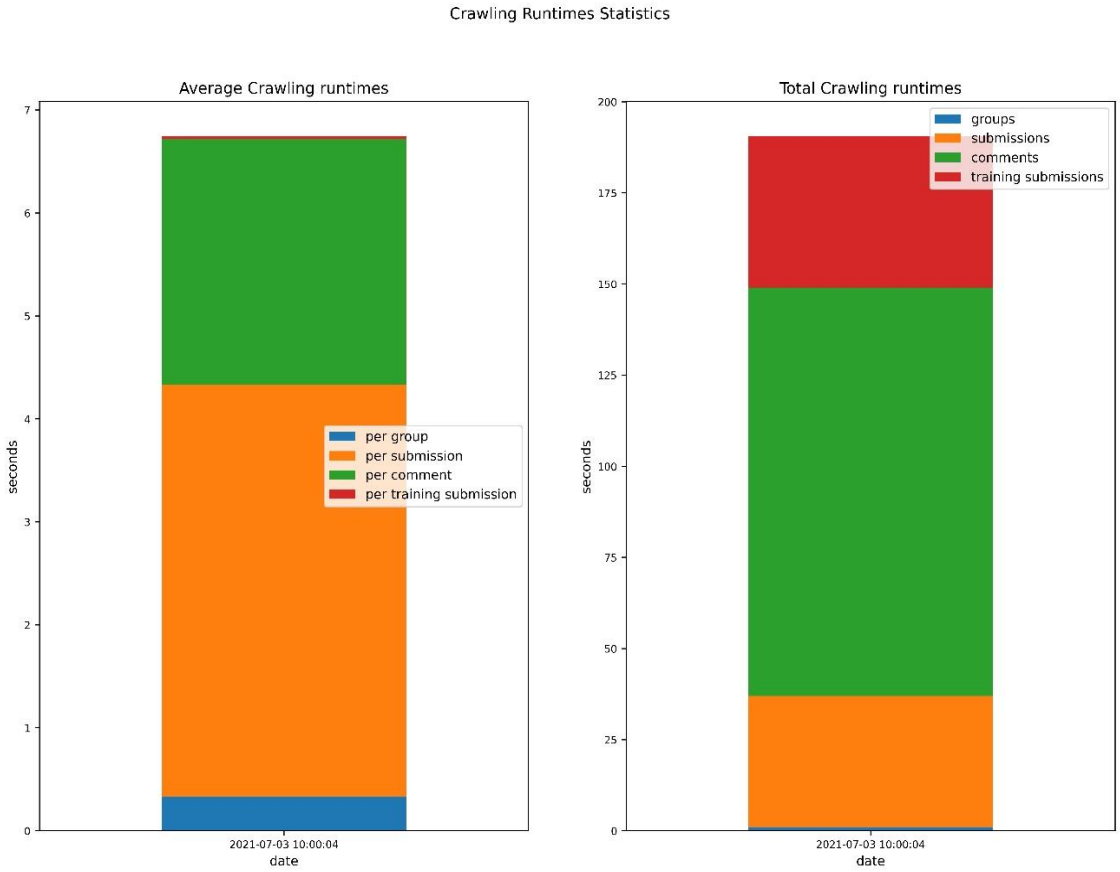

Figure 20, a stacked plot example of crawling runtimes after executing a crawling task from "Reddit.com".

- 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 includes 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 makes managing query syntax change and database communication much easier, along with 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" uses the JavaScript language to visualize the activity and influence graphs on the client web browser, and any requested segment of these influence graphs 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 better design of html elements in the same interface.

- JQuery

This is a popular feature-rich JavaScript library that gives the front-end of this application a more summarized syntax and work 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 web user interface run in isolated environments that is independent from the operating system of the host machine.

Another 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 in the source code of this technical solution. A single command will then instruct the docker engine to read the project configuration from the ".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 python interpreter installed on it. At last, docker spins up the web server for the user interface.

Docker also allow 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, databases of "mongo" and "neo4j"

Developers and administrator(s) of this solution might have the need to clear data from a certain database or plots on file system under debugging in development and testing environments. Or in production, where there might be a need to delete data crawled before a certain date due to privacy laws and regulations such as GDBR[11] rules.

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 through a task scheduler on the hosting machine.

VI - L. Future Improvments 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 easy to use 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 simultaneously. Which has the potential of reducing the amount of time that it takes to crawl big data chunks 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 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 that is nowhere to be found in the submissions database.

Concurrency and distributed computing can also help the algorithms of graph modelling to reduce their running time and being able to handle more data at 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 the 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 desired JS 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 give 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 by the help of cloud computing.

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 afterwards to give the administrator(s) a report about the health and performance of the system.

Security of this technical solution is highly important and needs to be reevaluated on periodic basis, 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.

[11] General Data Protection Regulations

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 patch are wiped out and a new set of records is created for these routes, this gives 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 simplicity and complexity of maintenance to achieve better performance with improved features of this technical solution.

CONCLUSION

We started this research project by establishing a ground data structure that fits into an entity-relationship model, then based on this model we defined a two-step algorithm for detecting and scoring influence between users of online social media; first by constructing the activity graph, then transforming this 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 social media, which is why we combined the 3 scoring techniques of influence together in a simple addition, which resulted in a wider range of score values, and higher variance between scores of different influences, such effect is desired for being able to distinguish and focus on influence edges with a significant high score.

Furthermore, a text classifier was tuned, trained, and built 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 users, which shows the possible field(s) of influence.

The process of text classification was designed to dynamically tune a text classifier for every daily crawling patch, providing the text classifier with an updated training dataset on periodic basis. The evaluation results of tuning this text classifier averages on about 0.7 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 between supervised and unsupervised machine learning, and we only have a rough idea on what topic a certain text might belong too 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 on the importance of each person in the graph from a score-dependent view.

The evaluation results showed a significant similarity between ranking by using the simple outdegree algorithm and the hub score from the HITS algorithm, these two ranking systems had the advantage of extracting people with high direct influence on their neighbors. While the betweenness algorithm was useful in detecting people who help spread influence by laying on as many shortest influence paths as possible. On the other hand, the auth score from the HITS algorithm was effective in both determining people 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 periodic daily basis, then use this data to detect, score and classify influence between active users on a targeted group or segment of a social media platform, and put all this information into an influence graph that is to be publicly available using a web interface, which 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 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 the scoring criterium, field of influence, or to find an influence path between two known users with the help of the efficient capabilities of the neo4j graph database technology.

The reliability and security of the technical solution was also evaluated carefully and multiple measures such as logging, caching and Digest User Authentication was 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 project, which is summarized in having an IT system that crawls data from online social media, then detect, score, and classify different influence types between users to produce a

series of informative, flexible, and constructive influence graph models on periodic time intervals.

ACKNOWLEDGMENT

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

The technical solution of this research had a great benefit from various online development communities and online documentation, and although the author of this paper has done his best to credit each contribution specifically in the following listings of references, some deficiencies might occur and please let the author of this paper know if this is the case.

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

REFERNECES

[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] "n-gram" https://en.wikipedia.org/wiki/N-gram (accessed July. 14, 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, 1th 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, 1th 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, 1th 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] “Digest access authentication” https://en.wikipedia.org/wiki/Digest_access_authentication (accessed July. 18, 2021).

[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 data" https://en.wikipedia.org/wiki/FAIR_data (accessed July. 21, 2021).

[23] "Convolutional neural network" https://en.wikipedia.org/wiki/Convolutional_neural_network (accessed July. 21, 2021).

[24] "Long short-term memory" https://en.wikipedia.org/wiki/Long_short-term_memory (accessed July. 21, 2021).

[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] "Reddit" https://en.wikipedia.org/wiki/Reddit (accessed July. 21, 2021).

REFERNECE WORK ON THE INTERNET

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

[29] 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).

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

[31] 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).

[32] 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).

[33] 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).

[34] 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).

[35] 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).

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

[37] 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).

[38] aljabear. “How to get the IP address of the docker host from inside a docker container.” Stackoverflow, 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).

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

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

[41] 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).

[42] Eyal Levin. “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 20.07.2021).

[43] 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).

[44] nikhilaggarwal3. “Create a directory in Python.” geeksforgeeks. https://www.geeksforgeeks.org/create-a-directory-in-python/ (accessed 20.07.2021).

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

[46] 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).

[47] 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).

[48] 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).

[49] 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).

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

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

[52] 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).

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

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

[55] 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).

REFERNECE OF TECHNICAL DOCUMENTATION

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

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

[58] “Neo4j Developer Resources” neo4j Developer. https://neo4j.com/developer/ (accessed 21.07.2021).

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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