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 had gained a great influence in different societies across the globe and on both individuals and different groups of people, 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, then score these influences and classify their area of influence. This project proposes a method for influence detection, then build a technical solution on top of it to enable analyzers of social media to have a visual and scientific understanding of the influence flow in social media.

Keywords: Social-media, influence-detection, influence-topic-classification, text-classification, graph-centrality, graph-node-ranking, IT-system-architecture, logging, caching, HTTPS-Basic-Authentication.

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 member. 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 depending 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, this opens from many useful applications that in multiple fields of society.

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 2016 presidential election in USA and the allegation of Russian interference is one example that can benefit from analyzing social influence on social media.

Furthermore, by mapping and visualizing social influence between social media users, we can speed up and improve the detection of fake news and other illegal activities on social media. By removing such damaging effect to any social environment, we can create a better and more healthy 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 in society.

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

# Introduction

Data from social media has a great potential in revealing the strengths of influences between network 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 establishing 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 functionalities between the most popular social media platforms, then produce a data model based on similarity between them. This data model is crucial to guarantee the reliability and flexibility of algorithms and technical systems that builds on top of this research; both reliability and flexibility are two sides of the same coin, as when basing the data model of the system on similarities between data provided by as many social media platforms as possible, we naturally widen the range of analysis potentials of any applying digital platform, making the solution more flexible and highly reliable and adaptable for social media analyzers.

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 such influences will be implemented to fit the different needs and purposes of the final application of analysis, regardless if the purposes are independent or can be tighten together in combinations. The desired result is a user influence graph where each node represents a participating user, while each edge between two given users representing the influence between these two users with respect to direction of influence and holding the strength and area of the influence whether it is in sport, politic, economy etc.

Following the previous effort, we evaluate the performance of the user 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 try to highlight the most interesting and useful features of the produced user graph and push its power of detecting influencers and their area of influence to the limit. The final test and evaluation process is a vital and necessary step to rely on the quality of the produced user graph model in any future analysis and technical solution.

This was a brief introduction of the upcoming research in a nutshell, but first let’s go through some interesting attempts in revealing user influence from social media.

# Related Works

Among the community of computer science, a wide variety of studies has focused on extracting information from the available social media platforms, and a big amount of effort has been dedicated to reveal influences between users to better understand the behaviour of individuals for many both commercial and non-commercial 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 to reveal specific influences and hidden behaviour patterns on different levels.

## Detecting and measuring user influence in 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 are dividing the influence evaluation models into 2 main categories; the first category is based on network topology which measure social influence of a certain user by considering his degree, shortest path, and random walk characteristics, while the second category bases the influence between users on their interactions through different activities organized in tree-like structures like submissions and multilevel comments. However, and despite the reasonably good classification and overview these researchers offers, their paper lacks some proven results of an experimental approach [1].

## Data-driven Influence Learning

Identify applicable funding agency here. If none, delete this text box.

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. 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. [2]

## Gathering data and Crawling alternatives

Most well-known providers of social media platforms offer

developers and data scientist multiple endpoints and ways to extract data from their platforms for development or analysis. Some research spots the light on this initial aspect of gathering data from social media platforms. One significant research is one that mainly describe the alternative of Pushshift Reddit Dataset by J. Baumgartner, S. Zannettou, B. Keegan, M. Squire and J. Blackburn. [3] The research paper offers an undirected but also claimed to be a more efficient and flexible way to gather data from the “Reddit” social platform than by using the official Reddit API endpoint.

It also gives an excellent brief description of the FAIR data 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.

## Classification of topics in social media platforms

As mentioned in the introduction, we are set to determine

the category of a detected influence between users, this opens up for the use of artificial intelligence for the purpose of classification between different topics where a certain user activity might fit in. In 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, a practical case study from an online health community was represented to give a good introduction of data pre-processing and cleaning, preceding to construct a reasonable mathematical approach for training and testing of a produced classification machine learning model.

Furthermore, the 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.

Another contribution of this research is the use of various deep learning algorithms to classify posted content such as CNN, LSTM and BiLSTM, all in which enable the researchers to achieve a promising F1-score of about 0.75 to 0.80 in topic classification.

The research has an excellent and well-performed evaluation and testing phase, where metrics of evaluation are carefully examined and explained in a good scientific approach [4].

## Choosing a study case social media platform

Determining which social media platform to crawl for testing and evaluation purposes is an important choice in the path to producing a user influence model that is flexible and useable in as many social media platforms as possible, this is why it is desirable to base the real life study case on actual real life datasets from a digital media platform that shares common user functionalities with as many popular social media platforms as possible, examples of such functionalities are post, 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 called “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 social news sites with Reddit included. The core of Steinbauer’s research lays in his analysis of subreddits, submissions and comments on Reddit which can serve as a foundation prove for why Reddit should be used in evaluating the performance of a user influence graph model and its ability to view the most influencing users in a social media platform, the reason behind this is that Steinbauer gives a detailed analysis on which subreddits seems to have the most of activity and also precedes in building into constructing a user graph model that helps showing which user has the highest influence based on users interactions through comments.

However, submission authors are not included in the dataset of the constructed user graph, making the user graph less reliable when ignoring the often-significant role of posters in generating discussions on social media. Another downside of Steinbauer’s user graph model is not using any other criteria than user interaction through comments, such as the upvote score or number of thread comments to a certain comment or submission.

Although Steinbauer has introduced a detailed overview of his evaluations and analysis’s results, there is still a big question mark on the technical details as no algorithms for constructing the user graph has been presented in detail.

# Chossing a case study social media platform to crawl for test and evaluation

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 below shows the top 11 most popular social media platforms in the last decade in the days from the beginning of 2011 to the end of 2020. []



Although Facebook is the definite leading social media platform, but there exists a competition in popularity when looking at the next top 10 platforms, with Reddit located in the middle of the popularity overview. A normal side effect of a more popular social media is the large amount of data users generate on such platforms making the platform slow to crawl and extract data although data from a more popular media often has a higher integrity. To keep a balance between data integrity and easiness in findability and accessibility, we will try to compensate between these 2 factors by choosing a medium popular social media platform for testing and evaluation purposes. Reddit is located at the middle of the popularity range in the figure above and can line up as a candidate in a study case for development, testing and evaluation purposes in this research.

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. []

A user on Reddit can create or join a group, make a submission on any group and comment on any submission or comment of 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 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 it 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 algorithms flexibility for future use on other social media platforms, 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 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 with having to worry about neutrality issues, but Reddit’s users also use a username that can be used to identify a person account with revealing their identity, which is a useful feature for general research and data analysis as we are working on this project.

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 that takes off programmers 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 many 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 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 request 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 along with its popularity between programmers who are crawling Reddit gives it an excellent record of ability to integrate with different products and systems that uses it.

* Reusability:

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

Based on the above four FAIR data principles and the user habits analysis between the most popular social media platforms, Reddit makes a good case study in the testing and evaluation process for us 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 data structures.

# Definiing a ground data structure

Flexibility of design is an important requirement of this research, as we aim to apply the influence detecting algorithms on as many social media platforms as possible, and although this might be difficult to achieve as a result of the wide variety of available social media platforms and their different user-functionalities, 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 there important to consider the desired results of this research before establishing a ground data model.

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 passive influence, an example of such influence is reading a newspaper where the reader gets influenced without adding any 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.

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, having this entity, makes it possible to study multiple social media platforms at the same time which increases the flexibility of design.

1. 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.

1. 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 the future. It 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.

1. Comment

The comment entity is very similar to the submission entity containing a body text, identification and author and location information. However, it has an additional feature which its ability to be a parent and/or a child of another comment. 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.



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

The attributes of entities open for more flexibility as we might 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 submission and comments. But also, attributes are generalized to match the most available details about these common entities 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 method for detecting and extracting social influence on a social media platform, the algorithms used in this method have multiple dimensions to reveal the strength and type of influences between different members.

# The ACTIVITY THREAD

Based on our previously established data model which satisfies the requirement for detecting activity- and interaction-based social influence, we can observe a clear hierarchy between the following four subject 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 other previous comments.

The figure below shows a small example of just on submission and it is thread comments, we can call this thread the activity thread as each single node in this thread represent an activity object that a certain user had performed, recall that activities can be either a submission which we set to be the root of the activity thread tree, or a comment which can only have one parent that is to be a submission or an another comment, and it can have multiple children as users can comment on this comment.



In the next section, we will see how we can use this activity thread to give birth to a weighted and directed graph that shows the interaction flow between users performing these activities, but first lets control the endless grow of activity tree by defining comments based on their level in the activity tree.

As comments can have their own thread comments, we shall establish a categorization of comments based on their level position in the hierarchy tree, where we can use 2 types of comments:

1. Top Comments: comments made directly on a submission.
2. Sub Comments: 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, as more sub comments than top comments can indicate that members are talking their time to read and react to top comment in addition to the submission which show a greater engagement of members than if they just keep themselves to writing top comments directly on the submission.

# 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 the parent activity to one of its child activities. Each edge represents one interaction between 2 users where target and source nodes are represented by their author/username, showing that the target user has reacted to an activity submitted by the source user, which in turn can be considered as an influence indication from a source user to a target user.

But knowing the direction of influence is not enough, as we wish to grade all known influences between users, so we can determine how strong a certain influence is in comparison to other influences in the user graph topology. For this reason, we will give each influence edge a score that shows how strong this influence is, many techniques can be used here for scoring influences, however, each technique have its own strength and weaknesses and the analysis system can only get stronger by offering multiple scoring techniques to be used in analysis, in this project we are choosing 3 different scoring techniques, each having some strengths and weaknesses depending on the use case of analysis.

1. Interaction:

This technique measures how many times a user B has reacted to activities performed by user A. The strength behind this technique 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 time in most cases, although the topic might be very interesting and therefore influencing for such users.

1. Upvotes:

By using the difference between upvotes/likes and downvotes/dislikes on the parent activity, we rely on the opinion of other reading users on the parent activity that is submitted by the influencer. The score is then given to the influence edge going to the user who had commented this activity.

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 whether the influence is supported or not by the society as audience. However, it is still possible the count of upvotes as a measure where 0 is the lowest score of an influence.

1. Activity:

The activity scoring techniques scores influences based on how many other activities descents from the parent activity in the activity tree.

As central people in society tends to influence the most people by generating a great number of activities such as politicians, this score can help reveal who are those central people and which users do they have an influence on. However, A downside of this scoring technique is that all child activities will get the same score which is the number of descendants activities of the parent activity, this downside can be compensated when users interact with each other multiple times in different comments and comment threads.

The figure below contains a graph built on the skeleton of the previous activity thread. It converts nodes from representing an activity to represent the user of that activity, while preserving the hieratical structure as it is in edges and their direction, it also introduces the 3 previous scores to be calculated making weighted edges.



After establishing our practical definition of influence and its strength based on different measuring techniques, we can digest the structure of the directed and weighted activity graph to produce a user influence graph having a unique node for each user and a unique directed edge from an influencer to the user being influenced, with weighted scores that is stored in each edge indicating the strength of influence based on the respective scoring technique. More details about how to produce and read the user influence graph is presented in the next section.

# INFLUENCE SCORE IN THE uSER iNFLUENCE gRAPH

After going through the activity thread tree and transforming it to an activity graph with directed and scored edges, it is now easier to digest such a graph to produce the desired output of a user graph where each node represents a unique user, while edges show a particular scored influence between 2 users with respect to direction of influence.

Looking at the previous activity tree and activity graph, we can count 2 participating users (User A and User D) who are generating 5 activity objects/nodes in total. To find out how much influence does “User A” have on “User D”, we go through the following steps:

1. Create a user node for each user in the activity graph.
2. Count the number of edges stretching from activities generated by “User A” and interacted to by “User D”, if the number is greater than 1, then an edge can be constructed from “User A” to “User D”.
3. The newly constructed edge will have three scores attributes that is calculated by summing up the respective scores from edges examined in step 2.

To clarify these calculations, we notice that an interaction score from “User A” to “User D” would be the number of edges going from activities generated by “User A” and interacted to by “User D”, while the upvotes score is the sum of values recorded on these edges and the same goes for the activity score.

1. This same process can be repeated to find the influence of “User A” on “User B” and its scored strength.

The result would be a new graph that visualizes the flow of influence between users, how strong the influence is and in which direction between users it is observed. The graph below is called the user graph and it is constructed using the detailed steps above based on the previous activity graph as an input.

The user graph does have 2 users with to influence edges between them; according to the interaction score, both users does have equal influence in each other, while using the upvote scoring technique tell us that User D has a slightly greater influence on “User A”, while the opposite is to be assumed when considering the activity scoring technique.



# TESTING OF INFLUENCE SCORing techniques

Previously, we have proposed a staged method that is meant to detect and extract influence between users who submits various activities such as submissions and multi-level comments on a social media platform. The output of this method is a graph where each node represents a user, and each edge indicates an influence from its source user node into its target user node. And because we are primally interested in the overall capability of such a graph to visualize the flow of influence between users, we are going to focus on the edges of influence between users as they hold the different scores obtained by different influence scoring techniques that were detailed in the previous sections.

For testing purposes, we will be using a dummy dataset with 2 main features:

1. The dataset is small to enable a logical and controlled transition from raw data about activities in a social network to a user graph with influence edge visualization.
2. The dataset is fictional but is inspired from direct observation of our case study Reddit and can be changed to reflect certain activities from a social media platform.

Testing the user graph model is essential to guarantee the reliability of performance of such model in a production environment with real data. In this test, we will focus on the influence scores stored in each edge of the user graph, to be able to see the entire dataset formed as a thread activity tree and as an activity graph please refer to appendix I

We are interested of knowing which scoring technique can help differentiate between all detected influences and their strength by looking at the distribution of all registered scores from each scoring technique (i.e., Interaction, Activity, and Upvotes) in a box plot in comparison with the frequency of each score value which is shown in the figure below.



The left column in the grid of the score plot above, shoes the distribution of interaction score values in our test dataset, where we can observe 4 single interaction between users, 3 double and 1 with 3 interactions giving a total of 8 detected influences. And because this dataset is not real, we cannot rely on this scoring technique to determine how good is this score in extracting and differentiating influence between users. The same goes for the activity scoring technique shown in the second column and the upvotes scoring technique in the third column.

However, when looking at the distribution of score values and their histogram frequencies, we notice an expansion of range of values and more normalized frequencies of activity and upvotes scores. This observation raises the possibility of combining the 3 scoring techniques in a simple addition performed on each influence edge and then observe the new variation of the newly combined scores compared to using a single scoring technique.

As shown in the fourth, fifth and sixth columns of the plot in figure above, we can observe that both the range of values has increased, and in addition there are more histogram frequency bars at multiple score values showing a higher achieved diversity of possible scores, the same observation is also possible when adding all the scores together to produce a new total score which the sum of interaction-, upvotes- and activity scores in each edge which is shown on the right column of the plot in figure .

Based on the previous testing results, we can conclude that each scoring technique is mostly depending 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.

In the next section, we will be using real life data crawled from Reddit, then take an overview on the calculated score values for each edge of influence between users. The evaluation process of scoring techniques is expected to reveal the true reliability of the user influence graph model and its ability to detect influences and differentiate their strengths between users.

# EVALUATION OF INFLUENCE SCORING TECHNIQUES

To investigate the reliability of our user graph model, we will be using Reddit as a case study for evaluation. The top 3 most rising submissions from each top 3 most popular subreddits has been crawled along with their comments and stored in the data structure we described earlier on in this paper.

After crawling and storing the raw data, the process of building the activity graph and user influence graph has been initiated to produce the 2 graphs show in the figure below



The activity graph above clearly shows the 3 defined types of activities: submissions in orange, First-level comments in blue and the second- and higher- level comments in red. Some submissions do not have any comments, other does only have first-level comments, while many submissions do house all multiple types of comments.

Understanding the activity graph before moving on to analyzing the user graph model which is built on top is essential for having an idea about which score is more useful in extracting the influence between users who submitted these activities, this is because the final influence scoring system in the user graph is based on the edges of the activity graph. i.e., in this case, we do have 9 submissions and 124 comments, 85 of these comments are first-level comment while 39 is thread comment on various levels, then the relation between top comments and thread comments is around 2:1 which indicates that half of the activities are submitted as a reaction to the submission itself and not on other comments meaning that if we use the interaction and activity scoring technique, then we are most likely going to discover the highest influence scores in the vicinity of the submission authors. However, since we have a significant share of around 31% thread comments from all comments, there is still an opportunity to detect influence between users who participate in activities lower in the activity tree based on both the interaction and activity score.

The analysis above is confirmed using a histogram plot of the interaction and activity score shown in the figure below where we observe that that most users interacts with each other only one time, and the activity score shows a concentration of low scores but still able to detect an influence strength from 1 to 7 between users. An interesting scoring technique is the upvotes based technique which in comparison to the interaction and activity scores, it varies in the range from 1 to 36 but with a low median of around 10.

As discussed in the testing process, it looks like no one scoring technique can be considered to be fully capable of indicating the strength of influences between users, but when combining scores in a simple summarization performed in each influence edge, we start observing a higher achieved variance of scores due to the effect of the level-independent upvote score.



Looking beyond the scores, we can now observe the topology of our user graph model where we were able to construct multiple influence edges between users and where the majority of those users are connected together using one-way directed edges as shown in the figure below. Where can identify 108 users and 121 influences between them.

The final result is a user graph model that displays each user as a separate node, and influences between them as edges which also tells us who is the influencing user and who is getting influenced by using the direction of this same influence edge, and not to forget the different 7 scores that has been calculated for each influence edge to indicate the strength of influence based on user interactions, activity levels or audience upvotes which can be used on their own and in any possible combination by summing them up for each edge. All these features working together increase the reliability and capability of such user graph model to reveal the most important influences between users.



# DETECTING AND CLASSIFYING THE FIELD OF INFLUENCE

In addition to being able to identify the different influencers on a social media platform, it is also an objective of this research to classify influences in multiple distinct topics that can be used as a feature of analysis. Knowing the type of influence between users is therefore just as important as detecting influences and determining their scores. To add such feature, we will be storing the types of the influence in its respective edge where the influence was detected.

We are here dealing with continuously changing data based on the text of both submissions and comments that varies over time because of new social events happening around the clock. Using a static classifier of text based on a fixed training dataset can therefore lead to misleading classification and hurt the reliability of our influence model. On the other hand, a dynamic machine learning classifier can be used to crawl text from a certain group on the same social media platform and use this text to train a text classifier assuming that all text generated by users in this group is related to a certain target topic such as entertainment, politic, economy, sport, and technology. This architecture of this classifier is detailed in the upcoming sections about the technical solution. But for now, we can assume we do have a reliable classifier that is capable of classifying text in between 5 different categories (entertainment, politic, economy, sport, and technology)

For the purpose of embedding the classified type of influence in the final influence edges of the user graph, we will be using a the same skeleton of the workflow for scoring influences from the activity tree, passing through the activity graph and finishing with topic classified influence edges between users. An example of this how this workflow is used to classify influence edges between users is viewed in the figure below. Where we start by constructing an activity thread tree from our dataset, then predict the influence topic carried in each edge of this tree by constructing a combined text from the root submission activity, the parent activity, and the child activity. If the parent activity is the submission activity, then only the parent and child activities are considered as an input to our text classifier.

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 will have the same classification, this is why the parent and child activities are also added to the input of the classification trial, which also helps detect any derailment in topic under the discussion or discussions that better suits the description of another topic than the topic of the submission itself.



The next step after classifying the topic of discussion between activities is to store this predicated topic in the edge between the parent and child activities, while labeling the activities with their submitting username to produce the activity graph. An example of such 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, we use the predict/classify method of our trained text classifier, which might output a classification into the sport category, and then store this output value in its respective edge between users in the activity graph.

The final transformation would be to some up all unique predicted categories from one user to another in the activity graph, producing the user graph model where each node is dedicated to a user and each edge contains an array of one or more topics of influence that a source user has on the target user.

Also notice that in the activity graph, an edge can only have one predicted topic and therefore a normal string type variable is enough to represent the topic of influence, but when constructing the user graph, multiple edges having the same combination of source and target users would be merged, and in results all predicated topics would accumulate and then registered by its distinct values in the edge of influence in the user graph model. This process is simulated in the user graph on the right of the plot in the figure above

Now that we have seen how similar the workflow of detecting, scoring and classifying influence is, we can proceed into merging the two processes to create a final user graph influence that can be used for influence analysis.

# TESTING OF INFLUENCE FIELD DETECTION

Text classification is the core challenge in implementing the previously explained process of topic classification based on the text of social media activities. To be able to have confidence in the results of influence classification, we need to have a reliable and well tested text classification machine learning model.

In this project we will be using a well-documented text classifier inspired from the scikit-learn.org community website [] 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 term in the provided training data to its occurrence count, followed by a transformation process that can benefit from using the Term Frequency-Inverse Document Frequency as a measurement tool, the last part of this pipeline is a text classifier that is to be built on top of vectorization and transformation to produce text classifier that can be used to categorize given text into different topics based on the provided training data. To gain more details about vectorization and transformation, please refer to .

Also, more details about the technical implementation of this text classifier are provided in the sections corresponding to this segment of the project in the technical solution.

To solve this challenge, it is possible to have a training dataset make up of categorized text which enables supervised learning and allow to predict or classify the topic of unknown text. And as most traditional machine learning models, the model ability to classify text with high accuracy is highly affected by the quality of its training data, 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, test its initial quality and tuning the text classifier for optimal results:

* Gathering training data

We start by gathering training data from different subreddits with a name applying to one of our categories that we are interested in under analysis, an example of a category collection can be politic, economy, sport, entertainment, and technology.

The top 100 newest submissions are crawled from reddit 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 subreddit with the target topic name as shown in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| 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 the plan explained above we obtain 1500 records in the best case after crawling reddit, these records would then be used for dataset evaluation, tuning of the text classifier, then into building of the user influence graph model.

This way of gathering training data from the same social media platform also make our model up to date in a continuous timeline, where training data 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 latest categorization of topic according the active community and not in fixed dataset.

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 the pseudo-random function that shuffles the data, this freezing allows us to have the same results after each run of evaluation, tunning or classification using this dataset, which comes in handy when trying to document the performance that can be achieved from the text classifier.

* Initial Evaluation of the text classification model

We start by testing the initial performance capability that can be obtained by using the crawled training dataset to build a non-tuned text classifier with its default key parameters. The training data is then partitioned into 5 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 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 increase the representation of every individual dataset record in the results that is calculated as the mean values of measured accuracies, precisions, recalls and F1 scores in the N=5 times the text classifier has been trained and tested which is represented in the table below.

|  |  |
| --- | --- |
| Metric | Measure Value |
| Accuracy | 0.696 |
| Precision | 0.703 |
| Recall | 0.696 |
| F1-score | 0.697 |

* Tuning the text classification model

When examining the results of the previous 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 the model using a collection of the key parameters of the involved algorithms, these parameters are stated in the table below along with their significance to the text classifier.

|  |  |  |
| --- | --- | --- |
| 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 to run our model with, we run a cross validation automated test using our defined test values. The result of tuning would be then be obtained as a combination of parameter values that gives the highest possible performance in term of the more reliable F1-score. The result of this tunning process in presented in the table below.

|  |  |
| --- | --- |
| Metric | Measure Value |
| Best score | 0.719 |
| vect\_\_ngram\_range | 0.001 |
| tfidf\_\_use\_idf | True |
| clf\_\_alpha | (1, 1) |

Using the outputted best score parameters, we manage to increase the F1 best score to 0.719, 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 is what is a good score under testing in artificial intelligence? The most sensible answer to this question is what we are trying to achieve from building the model? And as we are only seeking the use of this text classifier to detect and estimate the field of influence between users based on the text in their activities, a score of 70% can be considered to be satisfying to proceed into using this text classifier in building the influence graph model 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 next 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 turning 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 the table below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Category | Precision | Recall | F1-score | Support |
| Sport | 0.892 | 0.865 | 0.878 | 67 |
| Economy | 0.7 | 0.666 | 0.682 | 63 |
| Politic | 0.673 | 0.614 | 0.642 | 57 |
| Technology | 0.647 | 0.754 | 0.696 | 61 |
| Entertainment | 0.615 | 0.615 | 0.615 | 52 |
|  |  |  |  |  |
| Weighted Average | 0.712 | 0.71 | 0.71 | 300 |
| Macro Average | 0.705 | 0.703 | 0.703 | 300 |

The classification report indicates a weighted F1-score average of 0.71 with the highest F1-score for the sport category at 0.878 and lowest to the entertainment category at 0.615, the lowest score category might has been affected by the big interference of multiple aspects of the other categories into the entertainment business which can cause the model to falsely classify records that should belong to the entertainment category to other overlapping categories.

The confusion Matrix below shows the detailed classification of a total of 300 testing records where 213 records were classified correctly as expected, while 87 records were classified in the wrong category, 61 records of those with an incorrect classification stands between the 2 categories of Technology and entertainment on the one hand against politic and economy on the other side, leaving 26 incorrect classifications to the other 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 which tells us that this text classifier perform better when classifying general categories with little in common rather than classifying categories that might overlap.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Predicted (below) / Actual (Right) | Sport | Economy | Politic | Technology | Entertainment |
| Sport | 58 | 1 | 3 | 3 | 2 |
| Economy | 0 | 42 | 0 | 12 | 9 |
| Politic | 3 | 5 | 35 | 6 | 8 |
| Technology | 2 | 8 | 4 | 46 | 1 |
| Entertainment | 2 | 4 | 10 | 4 | 32 |

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 is it 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 test the performance of this text classifier with real life data from reddit to investigate how well this classifier is contributing to the objective of identifying different topics of influence in the user influence graph.

# EVALUATION OF INFLUENCE FIELD DETECTION iN THE USER INFLEUNCE GRAPH

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 2 users 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 the training dataset for the classifier which includes 1500 records distributed between topics according to the crawling plan detailed in table ?. all of the information attributes about the subreddits, submissions and comments are stored in the data structure visualized in the Entity-Relationship model in figure ?

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

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

Before preceding into studying the edges of the user 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 will 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 user influence graph, we visualize this information through 3 pie plots, one for each distribution measure as shown in the figure below.



The plot above proofs that our text classifier has been successfully able to identify 4 out 5 possible topics of influence between users. 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 the training dataset of the text classifier. For this reason, we will crawl the 3 newest submissions from 5 other subreddits where each subreddit is strongly associated with one distinct topic, 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 considered to be 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.



As we see from the plot above, 3 submissions were crawled from each of the 5 subreddits, and after building the user 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 considered 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 user 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.

We hereby can be confident of the capability of our text classifier to contribute into the process of building the user influence graph by classifying or predicting the topic of influence based on the submission text, combined with the texts of both parent and child activities of a certain edge in the previously explained activity graph, the edges of the activity graph would then be accumulated into a user influence graph with unique nodes representing users and directed edges labeled with the topic of influence.

# Introducing Centrality Measures

In search of identifying the top influential persons in a social 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 its favors the direct links between neighboring nodes or goes beyond neighborhood to examine the flow of many paths throughout the entire graph.

In our case, we have a user influence graph, representing users as nodes and influence between them as directed edges, for this case centrality measures have the potential to identify the power of each person in the user 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 degree 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 the 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 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 the 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 user influence graph, influence edges point to users indicating the influence of the source user in the target user, and because we wish to rank the top influential users in the graph, the outdegree centrality is a cleaner measure of revealing the power of influence from each user node, regardless of how many other users does influence this user.

To learn more about degree centrality, please refer to chapter ? in the book ?

Figure ? below shows an example of a user 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 others which gives them the lowest score of 0.

* Betweenness Centrality

Betweenness centrality is more concerned of the participating role of each user node in all the possible shortest paths between any given two users in the user influence graph. Following this principle allow us to rank each user based on his ability to carry influence from one user to another. The operation of the betweenness algorithm starts by listing all possible shortest paths in the given user influence graph, then counts the occurrence of each user in the connecting nodes between the source and target users, not to include these the source and target users in occurrence counting, each user 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.

The calculation of betweenness centrality is shown in the figure below, and it elects “User F” as the top influential user because this user lays on the shortest paths from both “User C” and “User G” to “User B” and “User E”, in the second place “User B” and “User C” gets the score of 3, while the rest of 4 non-central users gets the lowest rank of 0 because they do not participate in connecting other users in the graph.

To learn more about the betweenness centrality, please refer to chapter ? in the book ?

* HITS centrality – Auth and Hub with 10 iterations

The process of the HITS algorithm is divided into two symmetric but different calculations to 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 many 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 nodes gets the sum values of old auth scores given to the source nodes of ingoing edges, and 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, in this project we are 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 the actual steps of the HITS algorithm, please refer to chapter ? in the book ?

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 to each other 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 user influence graph are homogeneous, i.e., nodes does only represent different users, while directed links between them only represent social influence and nothing else.

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

Both graphs in the bottom of figure ? 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 user influence graph is a user that is linked by many different Hub users which are “User B” and “User C” in the right graph showing the hub centrality scores.



We hereby conclude that the auth scores helps detecting those users who are more likely to transfer influence between hubs in the user influence graph, while the hub scores on the other hand indicate the user’s ability to influence other users in the same user influence graph.

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

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

As we have already seen 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 og the user influence graph to understand the capability of each centrality measure that enables this analyzer to make the best decision on which user is to be considered the most influential. The table below summarized those previously discovered and discussed features of the various centrality measures.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 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 |

# CENTRALITY MEASURES VS. EDGE SCORE

Now that we have applied some centrality measures on the user influence graph, we can use the ranking results on each user in the graph to establish an understanding on which users are the most influential in the 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 of edge centrality, or it can take on the global scope of the graph by examining possible paths in it like the betweenness centrality. Scores of edge influences on the other hand are more personal as they indicate the source and receiver of influence and how strong each influence is, but still not to forget that in order to include a certain user in the user influence graph, at least one interaction between users has to be registered.

This leaves us to wonder what to use in order to find a good influential user, 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 object 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 hand if we would focus on the isolated user’s ability 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 users are to be elected as the most influential users between others.

An example scenario is given in figure ? below based on crawling the top 3 new 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 user nodes and less color transparency indicate a higher centrality rank.

Et bilde som inneholder kart

Automatisk generert beskrivelse

In the top right graph using the degree centrality to rank users, the top ranked user does not really have any strong influence compared to the second top ranked, 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 users position in the user influence graph. The same results are also reflected on the graph og HITS hub centrality but with greater separation between highly ranked nodes and those with lower centrality values.

On the other side, the top left graph uses betweenness centrality and as expected it has a tendency to give a higher centrality scores to those users who are able to connect other remote users in a shortest path, this is however not the case for graph using the HITS Auth ranking method, as users at the end of an influence path tends to gain a stronger rank, which is good if we are trying to find the most influential users in the user influence graph.

The table below shows the rank of the top 3 most distinguished or visible users of the 4 graphs in figure ? above

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 |

Both measures of outdegree and HITS Hub had yield the two users on top, however they elected different users in the third place, which is a good results because we are expecting both measures to have a certain similarity in detecting the most direct influential user in the graph, but also differ to reveal possible hidden information in the graph and help bring it up to the surface as in the third place where different users are elected using when changing from outdegree to the HITS hub measure.

When comparing the betweenness ranks to the HITS Auth ranks on the same graph, we notice a single common selection of user in the second place using HITS Auth, and in the next third place using the betweenness measure. 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 user has been elected in the second place using both the outdegree and HITS hub measures which shows that he is a good direct influencer, and the same user was elected in a group of third place users using the HITS Auth measure which tell us that this user 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.

Now that we have went through a practical example of analysis and cleared the common and uncommon features between edge scores and centrality measures. We shall say that in order to elect the right most influential user in the user influence graph, it is crucial to understand the ability and features of the used centrality measure and their impact on the final ranking results of users, and that those centrality measures should be studied hand in hand with the individual influence edges and their selected score, and type. This might sound 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 ?

# FUTURE IMPROVMENTS OF INFLUENCE GRAPH MODEL

Online social media is developing rapidly, and it is therefore a necessity to keep the previously developed model for user influence 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 user functionality to be used for detecting social influence for guaranteeing the capability and relevance of the user influence model in the future.

One example of possible user functionalities 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 on 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 features, because it is also possible to further develop the algorithm to implement a more advanced mathematical formula to calculate the weight of influence, rather than using the basic addition of scores with different types.

Also, 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.

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 or corresponding subreddit to learn from, or we can replace this dynamic text classifier with a static or external text classifier.

Another way of improving the process of topic classification based on texts generated by users is taking in account the entire discussion on a submission including all comments on it and then give every influence detected from this submission and its comments a single predicted topic.

Drilling down into our already implemented text classifier, we can also look for more improvement for accuracy and F1-score upon tunning av 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 users according to their importance in the user influence graph, 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 users, 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 user influence both relevant and informative.

# THE TECHNICAL SOLUTION

A desired outcome of this project is a system which can identify and classify influencers and their fields of influences 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 a easy to access and intuitive web user 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 in respect to the ER-model in figure ? to store it in an archive database where it can be used right away or looked up later on.

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 user influence graph, then store each graph in a separate dedicated graph database.

Another important segment of this technical solution is the integrated module dedicated to monitoring crawling activities and generating informative plots about each user influence graph like the distribution of different scores of influence edges and the percentage share between different types of influence. The generated plots are to be stored on the file system of the application host.

The final segment sets up a user interface in the form of a web server where constructed user and activity graphs can be retrieved from their respective database, along with the option of retrieving statistics about each graph and the initial crawling process. The next figure shows the system design architecture of this technical solution, this architecture is oriented to the actual dataflow from a social media platform to the final analysis tools provided by the user web interface.

Et bilde som inneholder tekst, parkering, elektronikk

Automatisk generert beskrivelse

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 JavaScript Object Notation, also known as JSON, which has proven capabilities in modern IT structure due to easiness in both readability and programmability.

After storing the crawled data on the mongo archive DB, this same data can know be retrieved and used internally to build the activity graph and user influence graph, 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 in our application such as calculating the node’s centralities upon constructing the user influence graph, also finding and retrieving a certain segments of the influence graph based on user-specified parameters such as score range or certain types of influence available through the web user interface.

# CRAWLING PHASE

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 back endpoint 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 ? earlier, the second process of the crawling phase is storing fetched data in the Mongo archive databases using an appropriate storing schema that maximizes both the efficiency of reading and writing data, and also makes it easier to a database administrator to manage and manipulate data 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 data from the same social media platform.

* Date of 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 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 time a crawling job is scheduled to run within one day, all data from this day will be stored in the same collection making it possible to arrange data in a daily timeline. The figure below shows the progress plan of a crawling job

Et bilde som inneholder tekst, parkering, skjermbilde, måler

Automatisk generert beskrivelse

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 database, this is to avoid any inconsistency and cascading effects that might a result of having an incomplete dataset because of a failure or lost of connection under the transfer of data from 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 crawling target, every class should at least implement 4 methods to perform the crawling and processing of data about target groups and its submissions and comments along with target training data for text 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 source code of this application. Later, an instance of this crawling class can be created to be a part of a script performing a scheduled mission where crawling, modelling, and other phases are performed in the sequence described by the draw of system design architecture in figure ?

The following diagram 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.



# GRAPH MODELLING

The next phase after processing and archiving the crawled data from a social media platform is to apply our modelling algorithm on this patch of data to produce an activity graph and a user influence graph. Both algorithms are explained in detailed examples under section ? and section ? along with centrality measures algorithms described in section ?

Although the user 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 user influence graph under development, testing or while debugging certain issues.

The next figure shows the detailed flow of different processes under 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, and then in step 5 and 6 an optimized text classifier is built to participate as a topic detector and therefore influence classifier in the separate processes of building both the activity-oriented graph and the user-oriented graph, and then storing each of these two graphs in their respective neo4j graph 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 following a wide variety of centrality measures like degree, betweenness and HITS.

Et bilde som inneholder tekst, overvåke, skjermbilde, skjerm

Automatisk generert beskrivelse

The source code that mainly concerns the modelling phase is to be found in the “modelling” module under the “classes” folder. In this module, we find the classes listed in the table below with its brief description og its purpose in the application.

|  |  |
| --- | --- |
| 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 topic 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 tunning 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 user influence graphs and saves the outputed graph to the respective neo4j database. |
| ActivityGraphModelling.py (ActivityGraph) | This class inherits from the Graph class and implements an algorithm for building the activity-oriented graph established in section ?, 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 user-oriented influence graph established in section ?, it also implements some needed methods especially for this algorithm. |

# GRAPH MONITORING & STATISTICS MEASUREMENT

Most graphs increase in complexity and the amount of information they hold when increasing in size, and our user influence graph is no exception, it is therefore important to monitor and digest the user influence graph to have a better understanding of its capability, reliability and integrity of revealing the full picture of influence flow between users.

Monitoring and viewing of continuous statistics about both nodes and edges of the daily user influence graphs is integrated into this technical solution 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 an pie share overview of social media groups and detected type of influence in the graph, these last plots were used as intended in the evaluation process of edges under section ? to evaluate the reliability of a user influence graph.

The plots are generated after the process of modelling and uses the popular matplotlib and pandas python libraries, then stores these plots as “png” images on the file system in a properly organized directory structure. These plots can then be retrieved by analyzers using the user web interface of this solution.

The user web interface also offers other methods for direct information retrieval from the user influence graph such as the centrality reports that shows a series of ranking orders of the different users in the graph according to the implemented centrality measures. And not to forget the possibility of viewing the underlaying activity-oriented graph which gives birth to the user influence graph and help explain it by visualizing the distribution of submissions, comments and thread comments in the crawled data using different node colors.

# DEPLOYMENT & RELESASE – USER 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 tool that includes many useful features such as viewing the entire user influence graph or the activity user graph, and finding an influence path between two users, or filtering influence edges according to a desired range of scores or 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 benefit its users.

Other features such as viewing the evaluation and tunning reports of each text classifier which is dedicated to a certain daily crawling patch is also included in the user web interface.

This web interface is founded in one script located in the root folder of the source code and called “ui\_web\_server.py” and as the name suggest, it spins up a web server after insuring that the underlaying database services are all up and running and therefore ready to receive queries.

For more information about how to set up a development, test and production environment of this project, please refer to the README.md file at the root directory of the source code.

# BUILDING A DRIVER CODE TO SCHEDUALE A CRWLING JOB

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 source code in the script called “reddit\_most\_popular\_subreddits\_driver.py”, this script can be used as a template example of the recommended way of producing a new crawling for this application.

The driver code starts with execution plan which defines the network name and submission type that distinguish this crawling job from others. In addition, it is recommended to define the different stages of this script in a stages array that enables the us to tell our driver code to jump or skip certain technical stages under development, testing or repair operations. An example of this exec 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 should then proceed by iterating over the configured runs in the exec plan, to try performing the following 4 technical steps of this solution:

1. Crawling

This is performed by using a registered crawling class in the “classes/crawling” module, in this case “RedditCrawlClass.py” is used to crawl data from reddit. Data provided through this class can then be written to the Mongo archive databases using the “MongoDBConnector.py” class which is customized for the needs of this application, and located in the “classes/database\_connectors” module in the source code.

1. User Modelling

This step uses the user modelling class in the “classes/modelling/UserGraphModelling.py” to produce the desired user-oriented graph which is explained in section ? earlier in this paper, then write this graph into the neo4j graph database dedicated for storing user graphs.

1. Activity Modelling

This step uses the user modelling class in the “classes/modelling/ActivityGraphModelling.py” to produce the desired activity-oriented graph which is explained in section ? earlier in this paper, then write this graph into the neo4j graph database dedicated to storing activity graphs.

1. Finally, the “Statistics.py” class in the “classes/statistics” module is used to produce the monitoring and statistical plots of this crawling run and store them on the host file system.

At every stage of executing the driver code, it is important to make sure that operations in the previous stage is able to 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 in figure ? 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 database during the process of crawling. And further on, step 5 depends on having results from all the above steps to retrieve monitoring results about crawling and model building.

In case there is a need to run only some selected stages, then simply provide the stages in the “stages” array included in the execution plan (exec\_plan) and an if statement should do the job of skipping or execution 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 submission from 5 pre-selected subreddits. All of these examples of drivers follow the same skeleton, and can be set to run at regular times by using a task scheduler on the operating system of the host machine.

# LOGGING & ERROR HANDLING

Handling errors and unexpected errors is vital for maintaining healthy IT systems. It is therefore important to log any exceptions triggered under the execution of system procedures and operations.

The technical solution of this project implements error handling and capture 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 directory of the source code. In other words, errors are not stored in dedicated databases but uses the file system of the project’s host machine.

Logging should be implemented according to the following steps 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” should be dated directories representing the actual date a crawling job runs on, i.e., “2021-05-20”.
3. Inside this date directory, multiple directories can be created for each network name or the name of the crawling job.
4. Finally, a directory for each submission types is created under the directory of network name and the error.log file is stored in 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 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 project instance, which results in an easier and straight forward debugging experience.

# 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 target group of a solution.

This technical solution is designed with an integrated database server for archiving data running on Mongo DB Server Instance and offers the system administrator the possibility to reuse previously crawled data from social media with respect to the date of crawling. Another important feature of having an archive for crawled data is to serve as backup data and in case of any inconsistency in the model graph data stored on neo4j, 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 data and modify them in the system databases. It is therefore important 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.py” and “Neo4jConnector.py” are provided in the “classes/database\_connectors” module in the source code and they are imported from both driver scripts for both writing and reading access and in the server script of the user web interface for reading access to these two database servers. Because we wish to establish a restriction on modifying any data from the script of user web interface, 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 user web interface to prevent any potential attackers to take advantage of the user 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 server of the web user interface, 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 user web interface by implementing a strong authentication like http basic, two-factor or one-time-password authentication. Http basic is implemented in this technical solution because of its simplicity and acceptable level of security when used together with HTTPS/SSL. For more details about how to use http basic please refer to section ????

# CACHING

The user 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. In addition, this user interface can connect to the archive databases to read crawled data directly, such as when enabling users to run and view results about the evaluation and tunning of the text classifier used in influence field detection.

Users can therefore experience high latency when using some routes made available by the web server that powers this user interface. To mitigate the negative impact of latency on user experience, file system flask caching is implemented on routes that has the potential of struggling with latency after receiving requests from outside users, an example route is the one that read text classification data from the mongo archive databases, then tune, train and evaluate the model, before providing the user with a full informative report about the text classification machine learning model.

Caching is controlled from the “.env” environment by specifying the following 3 parameters:

1. CACHE\_ON

A “False” value of this parameter switches the caching mechanism off, which is set as default in the “.env.example” file, and to use caching this parameter has to have the value “True”.

1. CACHE\_TIMEOUT

A default value of “0” tells the cache to produce cache records that never experience, and a positive integer indicate the number of seconds before cache records can expire after their creation.

1. 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.

The python “flask-caching” open source module is imported and used to generate cache records after initializing a cache instance of this module and mapping its configuration to the flask application instance as shown in the snippet code 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 snippet code above, caching is mounted to work on the route with the specified timeout configuration specified in the “.env” file 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 text 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 its time expiration, and then a cache record will either be renewed or updated and stored 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 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 results in consuming a lot of resources on the host machine and increases 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 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 all of the existing cache records and 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 re-running the driver code script.

The user 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 users, 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 solution.

# BASIC HTTPS/SSL AUTHENTICATION

This technical solution is designed to deal with data that is generated by users of 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.

The technical solution provides an integrated implementation of the http basic authentication to protect access to any route on the user web interface. The configured usernames and password are to be specified by the solution administrator in the “.env” files 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 hashed with a base64 hash function which easily guessed, and therefore http basic authentication should only be used over an HTTPS/SSL connection or on local machine. It is a therefore a good idea to implement a URL redirect rule that forces any received http requests to convert into using https to guarantee that HTTPS is used for strong encryption of the exchanged credentials between client and server.

To protect a route on the UI server, simply add the decorator “@auth.login\_required” above the route method you wish to protect using the http basis authentication over a secure HTTPS communication channel. 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 routes is important as an attacker can rerun multiple requests to this route to clear cache records more often than intended causing big delays in responding to requests from service users and by worst case a denial-of-service incident.

In this demonstration 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 other 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 decorator for the http basic authentication or consider implementing one of the two-factor or one-time-password authentication protocols if more scalability is desired for how gets access to the system UI.

# SUPPLEMENTARY TOOLS & FEATURES

The technical solution is provided with multiple supplementary tools and classes to provide additional helping features. These features are briefly described in the following:

* Crawling Runtime Register

An example feature is provided by the “RuntimeRegister” class which defines a structure for key parameter values on how long it took the crawling class 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 archive database under a collection named “crawling\_runtime\_register” and has a timestamp and marked with the unique parameters of the crawling job determined in the driver code. This register is 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 time measurements which is also provided to public users of this system via the user web interface.



* 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 is to be imported from the “classes/database\_connectors” module when needed in the operation of this application. This makes managing syntax change and database communication much easier because we ensure that we only have one edge endpoint to reach the data in a certain database.

* JavaScript Graph Visualization

To enable public users to access the product graphs of this system, a dynamic and browser-based visualization library called “vis.js” uses the JavaScript language to visualize the graphs produced by this application and any requested segment of these graphs such as a path between 2 user nodes on the client web browser.

* 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 frontend 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 to be noticed when setting up a development or production environment of this technical solution using the Docker instructions included in the “README.md” file that is attached to the source code. A single command will then instruct Docker to read the project configuration from the .env file, download the required docker images of the database and set up their containers, also a docker image with local code volume is constructed for the whole application with python interpreter installed on it. At last, docker spins up the web server for the user web 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 the current production environment.

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

# FUTURE IMPROVMENTS OF THE TECHNICAL SOLUTION

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 improvement 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 no where 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 improvements of the user graph model can also 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 any JS library for plotting data directly on the client side of the UI instead of using matplotlib on the server side. The user interface itself does not have to use the bootstrap library, and the front-end can be self-designed or it can use any other available 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 user interface of this technical solution. This might be a challenge regarding resource management, access control and security issues, but many similar projects have lately been successful with the help of cloud computing.

Error handling and logging mechanisms can also 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 us a report about the health and performance of the production environment.

Security of the technical application is highly important and needs to be reevaluated on periodic basis, the http basic authentication should not be used over an unsecure http connection, and a better encrypted authentication can be used, such as 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 an technical application.

Finally, we can consider improving the caching system of the UI by using another type of cache than filesystem cache, such as Redis 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 dedicated a cache system that is customized to the need of this technical solution. In addition, we can implement the possibility to create, remove or renew a cache record for a particular route on the UI, as in the current caching implementation cache records are wiped out and a new set of records are created for all earlier graph models, this means that caching time will increase if the system continues to crawl and process more data, but with a selective caching technique we can just create or renew the cache record or a target route with all of its unique parameters.

We have hereby discussed some potential improvement that can help in increasing the reliability, availability, and security of the use of this technical solution. It is also very important to evaluate these and other possible improvements in the future with its the gained outcome value, and not to forget the importance of keeping a balance between simplicity and easiness of maintenance toward better and more features of this technical solution.

##### ACKNOWLEDGMENT

This project has benefited greatly from previous 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 welcome to contact the author of this paper for any clarification.

The technical solution has also had a great benefit from the developer’s community at “stackoverflow.com”, and although the author of this paper has done his best to credit each contribution specifically, 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 of this project at <https://smia.uis.no>.

##### CONCLUSION

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

The user-oriented graph viewed users as different nodes and visualized the influences between them as directed edges indicating that the user located at the beginning of an edge is the influencer and the user being influenced is located at the target of an edge, those edges has also been scored by 3 different scoring techniques; interaction, upvotes and activity, each of these scoring techniques contributes in understanding the full picture of influence between users, which is why we combined the 3 techniques together in a simple addition operation which gave us a satisfying result of wider ranges of possible scores and higher variance between scores of different influences, such effect is desired for being able to distinguish and select influences with significant high score.

Furthermore, a text classifier was tuned, trained, and built to predict the topic of an influence based on the text of the source- and target activities in the activity-oriented graph together with the submission text. This text classifier was integrated into the algorithm of building the user influence graph to classify influence edges between users, which shows the field of influence. The process of text classification was designed to dynamically tune a text classifier for every time a new data crawl is performed, 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 since we have a classification problem that requires a mixture between supervised and unsupervised machine learning, because we only have a rough idea on what topic a certain text might belong too only based on the kind of subreddit it was posted on.

Centrality measures such as the outdegree, betweenness and HITS were calculated on the produced user 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 user in the graph from a score-dependent view. The evaluation results showed a great 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 users with high direct influence on their neighbors, while the betweenness algorithm was good in detecting those users who help spread influence between other users 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 in addition can reveal users that is most likely to be influenced by other users.

To apply these learned theoretical measures and newly developed algorithms to real-life, a technical system was developed during this project. The 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 influences between users then put all this information into a user influence graph that is to be publicly available using the user web interface, which also allows for retrieval of more information about each daily influence graph such as crawling time, topic share in influence classification, then histograms boxplots of score distribution, and not to forget the useful centrality ranking of user importance and the possibility to view the tuning results of each built text classifier. The user web interface also offers a series of tools to filter the graph based on score or field of influence or to find an influence path between 2 known users with the help of the uprising capabilities of neo4j graph databases.

The reliability and security of the technical solution was also evaluated carefully and multiple measures such as logging, caching and HTTPS Basic 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, practical and technical pieces together, we construct a big picture that satisfies the objective at the start of this project, which is summarized in having an output system that crawl data from 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.

##### References

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

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

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

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

[5] T. Steinbaur, ‘‘Information and social analysis of Reddit,’’ inProc.TROYSTEINBAUER CS. UCSB. EDU, 2012, pp. 1–12. [Online]. Available:http://snap.stanford.edu/class/cs224w-2011/proj/tbower\_Finalwriteup\_v1.pdf

**IEEE conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove template text from your paper may result in your paper not being published.**