1. Introduction

Making the predictions for such users related the account or user is similar to making the predictions of posts for that user. Hovewer taking up some processess to filter like mathematical formulas and graphs are kind of similar follow up at logical part. We are going to se it furtherwords in this Documentation

This document is for educational purpose and fully provided officially for community collaboration and taking up algorithms steps socially. You are viewing public copy of implementation of mathematical steps and logical workflow about this steps. However this is also one of algorithms that issued under ©2022 of Spade Org. Enjoy!!

Specifically, if we are talking about the predictions of accounts, it can be done by so many methods,

- **Geographically –** Suggest users which are nearabout there areas and are in trending era on this platform
- **BackLink Approach –** Suggest the users which are near to you in terms of social life, for example you and userB are good friends at XYZ platform. If userB appears on Spade, it'll be a good prediction to be put him/her to your recommendation list.
- **Content Based –** The users which are already on This platform and preforming nice/posting nice spacks, it'll be a good prediction.

There are also several more methods, used by so many social media platforms to keep you engaged and surprised by impossible seeming results.

Spade Trending Folks Predictions – Behind the scenes

As on now, we decided to take the recommendations according to 3rd method (**Content Based**) filtering, Why?

- we have sufficient data to work with and user's social behaviour can be monitorized by platform itself properly.

If we think of taking down first approach (**Geographically**), you see, in order to gather data users have to give permission to access their location. Which is kind of arggghhhh.... moment for most of users. As a result the system have to work with less data.

Working with insufficient data leads to in-consistent results and can't predict right results. For 2nd approach, users still have to link most of their social media in order to give right recommendations. Which is again headache task to login from socail media accounts. Like me, i don't even remember my password of instagram. I always click on forget password option when i have to relogin in it xD

So, now you are aware why choosing 3rd option takes the sense rather then it's above options, ok Talks too much let's dive into Tech Part!!!

2. Implementation Workflow

As of now its clear that we are dealing with data with our platform. Which leads to happiness to system because it has access to resources of it's own platform. Wow!! now Let's get serious.

So if i ask you a question who is trending in your group? Give me top 2-3 names.

What you consider to sort these 2-3 peoples out? Not the person itself, maybe not the looks but you are sorting by their qualities they have got. You determine everyones character by what you see of them.

So it's now getting clear, system can determine the accounts by the data its got like how often he posts how people at platform reacts to that post. What comments are passed over to actions. How much your account got reported.

So whats your answer of above question? When I'll ask it you at college/school?. Maybe you sort them by how much they are goot at studying/learning/practicals. Which is right depending upon asking environment.

Same goes here, we are sorting the accounts bsaed on how many quality spacks are being contributed by account. This is one of the factors in out initial filter.

Getting technical:
Now, on platform there are two types of users.

1. Authenticated (Logged In) – System identifiable
2. UnAuthenticated (Logged Out) – Anonymous

if the user is authenticated its easy to get recommendations because we can connect links including the user's account itself.

If user is not authenticated, we have to show them trending accounts, like trending account s all over the platform – *Can be region specific*

In this section we only cover General Trending Folks finding algorithm, as second one is complex and decised to be confidential now.

3. Calculating Actions

Because, we are sorting according to spacks. The biggest factor to consider is obviously number of likes it got and number of dislikes it got / or / how much spack is being reported.

We can generally call it score or normalized number of likes, which can be expressed by,

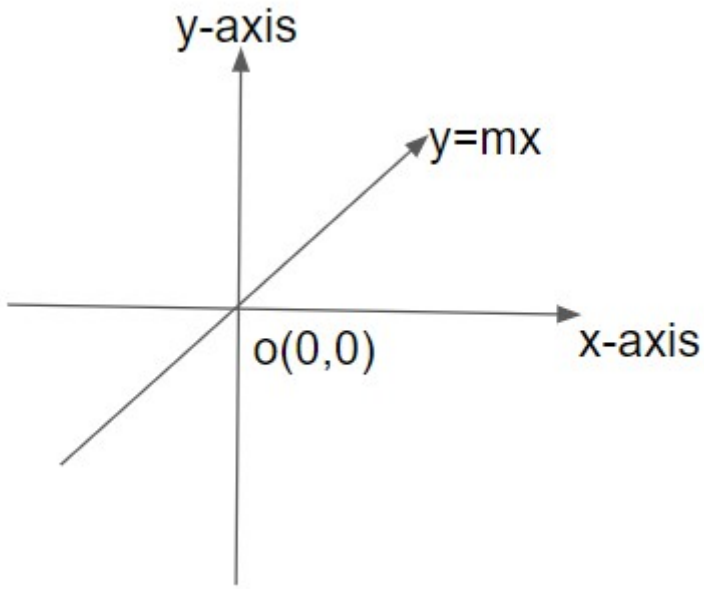
$$score = numberOfLikes - numberOfDislikes$$

We take some pre-cached spacks in monitor and calculate the score of each spacks and filter them accordingly

After we do several internal filters to take the score almost neutral position./ native value

Afterwards, we need some graph to represent the growth of posts with respect to time.

How about linear progress of spacks
Ex..



Consider y axis as trendness and x – axis as time in seconds. Hmm what a simple graph to work with!!

(with considering the m-Slope as score of that spacks)

But it doesnt' apply in real application beacause it means the older you posted the spacks, the more trending your account gets in picture. Which beats the purpose.

So, we now clearly see that time is not supporting factor but dividing factor. Let's try then

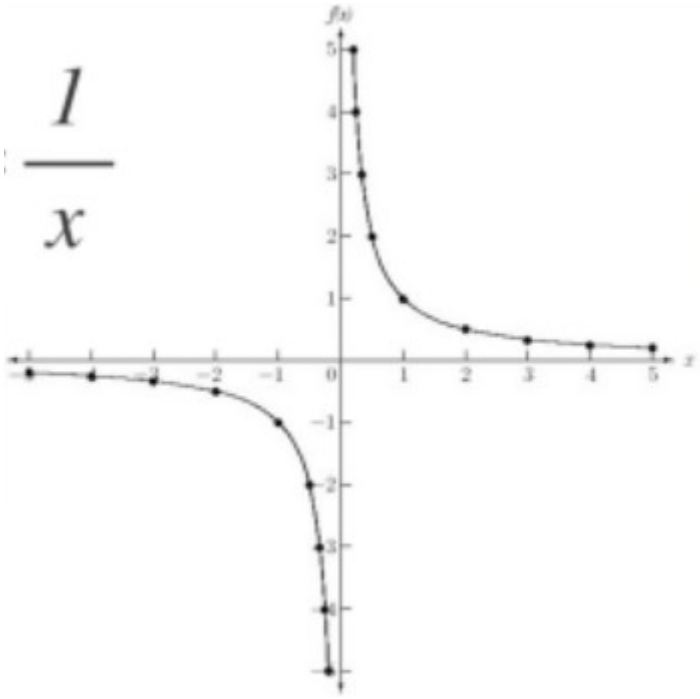
$$trendness = score / time$$

Which if we put above assumed variables, leads to,

* $y = m / x$

* $y = m\, x^{-1}$

equation. Which if we visually represent then...



Ok, Ignoring bottom left graph its looks like we got the cirrect graph that fits our need!!

Which is correct in majority of the cases but the only problem with this graph is that if the posted content got boosted after nautral journey. The graph still slows down and doesn't show the BOOM created in growth.

Its in the system tho. But as primary filter. Because as we discussed we can't trust it at all because it can't detext the boom created in between easily.

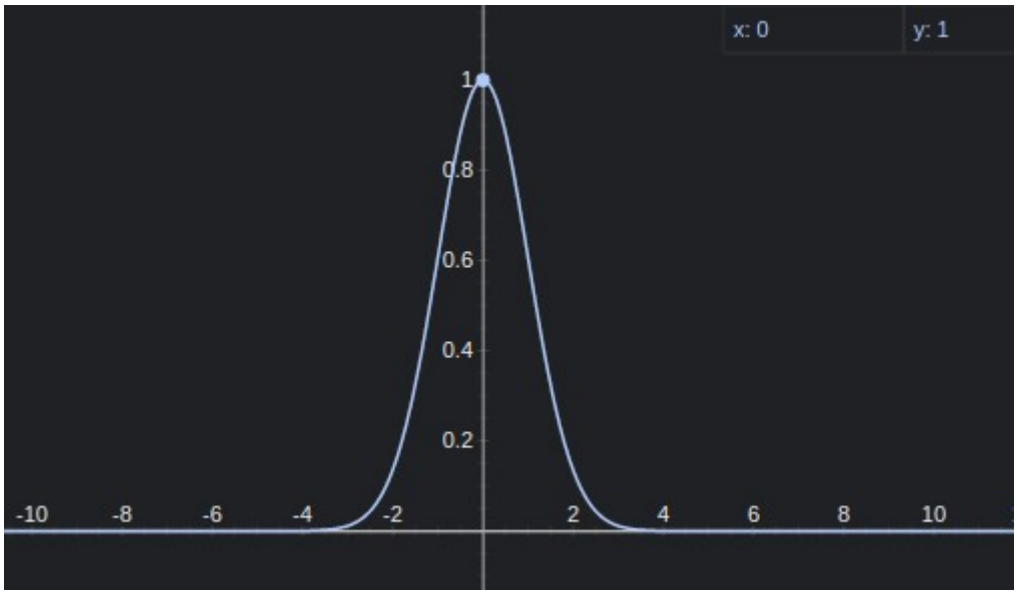
OK!!, finally we need a bumpy graph which can be responsive to bounce growth of spack posted.

If you are familiar with ML, you have heard about Confidence Interval (CI), we need the graph just like it .OHH, gosh what a sexy curve just as we need in this case fits perfectly.

Rather then taking the graph of it, we took the formula to sort the final list according to how confident we are about the recommendations.

BTW, the equation of that bouncy curve is similar to Eqn,

$$y = e^{(-(x)^2/2)}$$



Its a little bit sharp tho, but by adding some constants it can be overcome easily!

Notice: *we divide the* $-x^2$ *with 2. which division factor is not fixed. However this represents the broadness of the curve. Or how wide the curev is gonna be.*

$$y = e^{(-(x)^2/BROADNESS)}$$

As we can observe, the maximum value can be reached is 1. but at time t=0, we dont want to make spack's trendness maximum (when its just posted), also we are ignoring the score variable.

So lets move it Right a bit as our need. I am taking here RIGHT_SHIFT (**RS**) constant which you can assign any value fits your application

$$y = e^{((-(x-RS)^2)/BROADNESS)}$$

Ok finally we just need to consider the **score** in order to maintain that bounce effect. Of graph which in our case fits perfectly if we multiply it with time(**x**)

We have to also keep in mind that the multiplying term (score) doesn't make result exceeds 1 as its the highest value yielded by graph, so we take its inverse, as end we got value between 0 and 1) but now we are considering score as negative factor, however we want to conosider as supporting one, also if we take multiplying factor as 1/score. What if spack just posted?, score=0, so multiplying factor became infinite. OMG glitchhyyy!

So the final Equation comes in as covering above mentioned problem is with smoothing with high constant,...

$$trendness = (1/(EL - score)) * e^{(-(x-RS)^2/BROADNESS)}$$

* EL is effective number of likes / platform dependant.

Almost done, the catch in above equation is if effective number of likes became same as score, denominators spoils the game. To cover up, we are adding bound

$$trendness = (1/MAX(1,(EL - score))) * e^{(-(x-RS)^2/BROADNESS)}$$

Which yeilds nearabout the right results as expected.
Now because we sorted the posts, its easy to find associated account to that post(author).

We can use the AI model (Like, Neural Network) to train and predict the data as well, but this approach consumes So much memory + we have to be rich at data providing othervise it can be predict wrong results. Using Algorithmic apporach joints the best way this platform needs now.

Thanks