DISTRIBUTED SOCIAL NEWS

by

STEFANOS CHATZAKIS
URN: 6481123

A dissertation submitted in partial fulfilment of the
requirements for the award of

BACHELOR OF SCIENCE IN COMPUTER SCIENCE

May 2020

Department of Computing
University of Surrey
Guildford GU2 7XH

Supervised by: Sotiris Moschoyiannis

I declare that this dissertation is my own work and that the work of others is acknowledged and indicated by explicit references.

Stefanos Chatzakis
May 2020

Abstract

This Report describes the design and implementation process for a web based system that provides it's users a personalized view of the top social news depending on their interests. Moreover, in order to achieve this, the report will also focus on implementing some of the most popular ranking algorithm which will then be compared with each other in order to select the most appropriate for the project.

This report will loosely be split in two parts. The first being understanding the strengths and weaknesses of each ranking algorithm and comparing them to find the most suitable. The second part which will focus entirely on the implementation using an agile development methodology.

All in all the aim of this report will be to explain the challenges I face during the comparison process as well as the development process.

Acknowledgements

First of all, I would like to thank my parents Fedon and Katerina for all the sacrifices they have made for me to have this opportunity today, but also their continuous support and motivation. Furthermore, I would like to thank my sister and my close friends who have helped me cope with the difficulties I have faced during the past three years.

Contents

List of Figures

List of Tables

Glossary

T	The time stamp used in the ranking algorithm.
a_i	The score of each interest
$\sum_a$	The sum of a

Abbreviations

BER	Bit Error Rate
BPSK	Binary Phase Shift Keying
BSC	Binary Symmetric Channel
DCT	Discrete Cosine Transform
ECC	Error Correcting Codes
FEC	Forward Error Correction
JPEG	Joint Photographic Experts Group
MPEG	Moving Pictures Experts Group
SER	Symbol Error Rate
SNR	Signal to Noise Ratio

Chapter 1

Introduction

1.1 Background

Current social news sites try to produce a list of current important news items by applying algorithms and voting by the community. While more decentralized than a traditional publication, the system still relies on a select few(moderators, administrators) to safeguard it from manipulation (spam, vote-rigging) while maintaining the culture of the website. Further, by providing a centralized view of the current news, they necessarily average the preferences of their members instead of addressing the needs of each one based on their voting patterns. This project aims to develop a new type of social news aggregator that allows each user to see a personalized view of current events while allowing them to customize the algorithm by which the items get selected.

1.2 Aims & Objectives

The solution, which will first be theorized and made into an algorithm, will first be comprised of the combination of the most critical aspects of the most popular sorting and ranking solutions out there while tailoring the final solution to one that fits a social news aggregator.

1. Develop a Functioning website with users that can create, edit their posts and view all of the users post ranked by timestamp.
2. Compare the most commonly used algorithms, critique the positive and negative attributes and theorize, which parts of each algorithm should be implemented in the final implemen-

tation.

3. Compare the relevancy of the results after the final version of the algorithm has been implemented.

1.2.1 Project Limitations

Since the development of this project was initiated the goal was to create a testing platform on which to compare the results of the implementation of the final algorithm in a real-world environment. Thus, the scalability of the project as a whole was not taken into consideration. Having that said, the issue of scalability would occur not because of the algorithm itself, but due to the reason that the software developed does not address many security issues as well as correct optimization so that the system would be able to hold on with more users.

1.3 Software Development Life Cycle

Software Development Life Cycle (SDLC) is the process that a software project follows to develop and test the system. A Software Development Life Cycle can be thought of as a master plan that describes how the the development, testing, enhancement and maintaining processes should occur during the implementation of the software (Stackify, April 2020, p.1).

The goal of the SDLC is to produce a software with the highest quality at the lowest possible cost. In order to achieve this, SDLC provides six phases that are consistent with the most popular methodologys such as waterfall, spiral and Agile (Stackify, April 2020, p.1). The SDLC phases include:

- Requirement analysis
- Planning
- Software design such as architectural design
- Software development
- Testing
- Deployment

The most traditional method for software development is the Waterfall methodology. Waterfall, delivers the above phases in a linear fashion, meaning that each phase can begin only after the previous has been completed. Moreover, thanks to the implementation of the phases, the waterfall methodology becomes a fairly straightforward process. However, a process can never be repeated unless a critical error occurs that requires the revision of a previous phase. This makes it incredibly risky to use in a final year project development environment since fully following the methodology, could prove difficult to address the arising errors that are bound to occur.

A more common methodology in the recent years is the Agile development approach. This approach takes the phases previously mentioned and creates the so called sprints. The sprint's duration usually is a couple of weeks long with pre-specified goals set at the beginning of each sprint. Each sprint can be thought of as a many smaller implementation of the waterfall approach, since the phases are consistent with both methodologies but since agile is comprised of sprints, during each sprint each phase will re-occur.

1.4 Chosen Methodology

After considering most of the commonly used methodology and due to the fact that I didn't have a lot of time to chose a methodology that was too complicated or had too many restrictions, I chose the agile development methodology using the Kanban framework. The goal of using this framework is to aid the developer in the categorisation and prioritisation of the work and tasks. Furtherore, the agile methodology was chosen since before the development, the technologies used were previously known. Having that said, using the agile approach the development will be broken down into small increments which will encourage the mastery of the technologies used after each sprint. Lastly, the sprints will allow for overall better testing and feedback, since they will be received after each sprint.

1.5 Report Structure

The report structure will mirror the planning and development process as it occured. First by reviewing and constructing the ideal ranking algorithm for a social news aggregator following the development of the web application which will implement the constructed algorithm.

1.6 Summary

Overall the introduction chapter has portrayed overview of project's definition, flow and goals as well as addressing how the project will deal with the different phases of development through defining the methodology.

Chapter 2

Literature Review

2.1 Technologies Used

During the entire design and implementation process, the decision was made to use the Python programming language. This choice was made originally for the ease of implementing algorithms in a scripting language like python. However, while progressing from the testing phase to the development phase, a dilemma was faced. Either to re-implement most of the progress I had made to a language that was more web and browser-based like JavaScript or, to create a hybrid of both front-end and back-end schemes. After some research online, Django was used, a high-level Python Web framework that encourages rapid development and clean, pragmatic design.

2.1.1 Django Framework

According to the official Django website, "Django is a framework that was built as a tool for front-end developers that needed a simple way to bring their ideas to life without the need of a back-end developer that handles processes such as creating and connecting the server-side with the client-side as well as the creation and handling of the system's database (Django, July 2005). Furthermore, Django is a high-level Python framework that utilizes SQLite as a relational database. Moreover, Django uses an MVT pattern similar to the more widely used MVC that frameworks like Ruby and Ruby on Rails uses.

2.1.2 Atom

Since the whole project was entirely in Django, it meant that all the code was compiled through the console. For the text and file editor, Atom was used for its familiarity and simplicity. Moreover, for testing and displaying the application, Google Chrome was used.

2.2 Existing Solutions

2.2.1 Reddit Rank (Hot Sort)

After conducting research, it was clear that one of the most popular social news aggregators currently is without a doubt, Reddit. Reddit, although it has now switched to a different ranking algorithm, since the end of 2010 but have since made their old ranking system available to the public. Their algorithm explained in the purest form takes many parameters such as the timestamp of a post, the difference between a post's upvotes (likes) and downvotes (dislikes) and inputs those parameters in a formula that outputs a final rating that dictates a post's position compared to others.

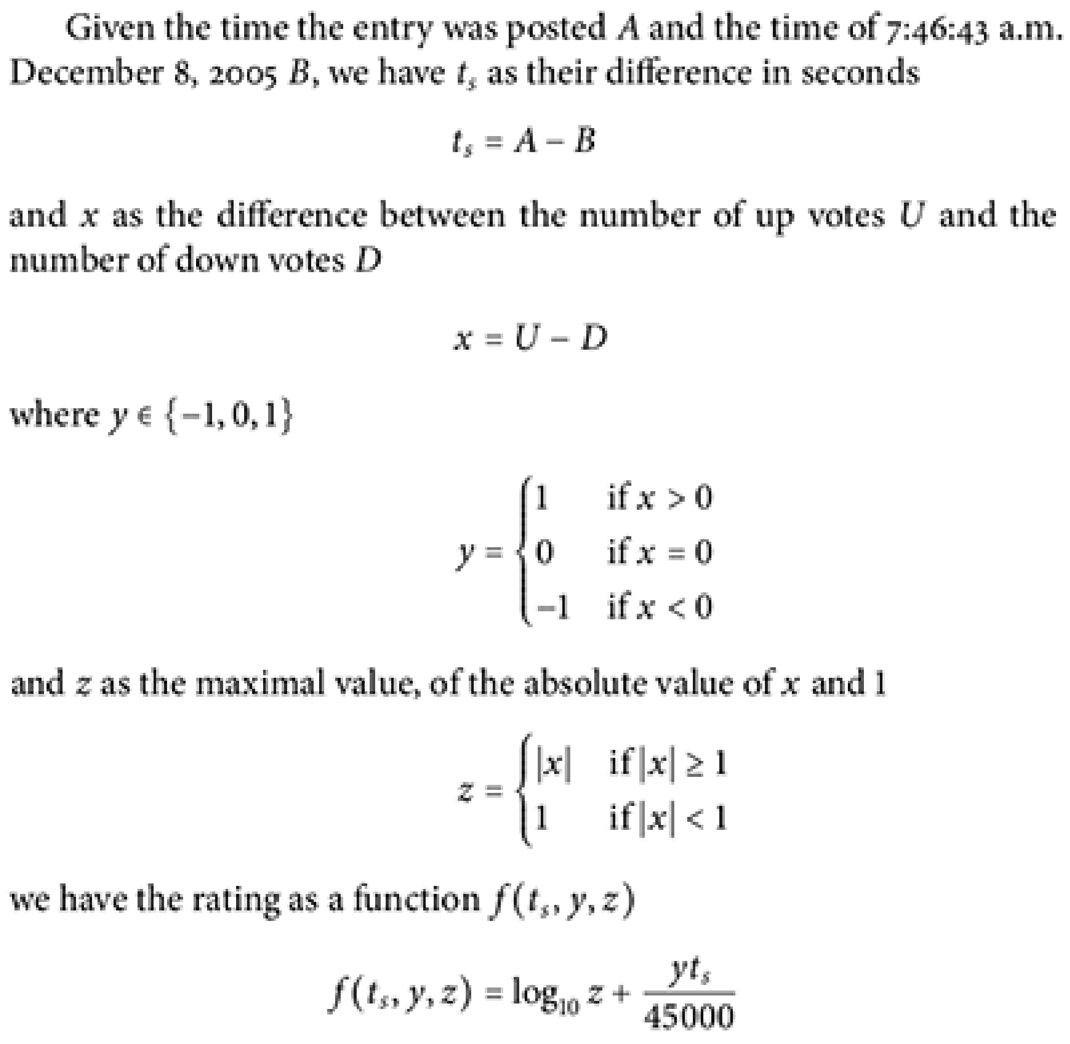
2.2.1.1 Weaknesses

- An issue which is absent currently Reddit, but present in a lot of websites that take average rating as a rating attribute, is the following: Average rating works fine if you always have a ton of ratings, but suppose item 1 has 2 positive ratings and 0 negative ratings. Suppose item 2 has 100 positive ratings and 1 negative rating. This algorithm puts item two (tons of positive ratings) below item one (very few positive ratings).

2.2.1.2 Conclusion

2.2.2 Hacker News Rank

Hacker News is one of the most popular social news aggregator targeted towards developers and provides its users with mostly technology related news. Their ranking consists of three parameters, penalties, votes and age. The most impactful parameter on the formula is the penalty. The penalty's value is determined by the use of blacklisted words such as "NSA" which



Given the time the entry was posted A and the time of 7:46:43 a.m. December 8, 2005 B, we have t_s as their difference in seconds

$$t_s = A - B$$

and x as the difference between the number of up votes U and the number of down votes D

$$x = U - D$$

where $y \in \{-1, 0, 1\}$

$$y = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}$$

and z as the maximal value, of the absolute value of x and 1

$$z = \begin{cases} |x| & \text{if } |x| \geq 1 \\ 1 & \text{if } |x| < 1 \end{cases}$$

we have the rating as a function $f(t_s, y, z)$

$$f(t_s, y, z) = \log_{10} z + \frac{y t_s}{45000}$$

Figure 2.1: Reddit Ranking Algorithm in mathematical notation

drops the story rapidly in the ranking. In order to keep the top stories fresh, Hacker News also issues a severe penalty on stories that reach 40 comments. The impact of a penalty can be calculated with the scoring formula since if an article gets penalty factor of 0.4, each vote will now count as 0.3. However, a factor of 0.1 is equivalent to each vote, counting 0.05. Meaning that although a penalty factor of 0.4 would drop an article 66% faster than usual, a factor of 0.1 would drop an article by 3.6 times than normal. In outline, each item is given a ranking, and the articles are sorted according to the ranking. The simplistic way to think about ranking is the number of votes is divided by time, so more votes results in a higher ranking, but the ranking also drops over time. The votes are raised to a power less than one, while the time is raised to a power greater than one, so time has more effect than votes. Some additional penalties also may be applied to the ranking.

$$rank = \frac{(score - 1)^{.8}}{(age_{hours} + 2)^{1.8}} * penalties$$

Figure 2.2: Hacker News Algorithm in mathematical notation

2.2.2.1 Weaknesses

1. Wall-clock hours penalize an article even if no one is reading (overnight, for example). A time denominated in ticks of actual activity (such as views of the 'new' page, or even upvotes-to-all-submissions) might address this.

2. An article that misses it's audience first time through, perhaps due to (1) or a bad headline may never recover, even with a later flurry of votes far beyond what the new submissions are getting.

2.2.2.2 Conclusion

Overall, Hacker News's algorithm is quite simple, thus making the implementation of something similar not that difficult especially since it takes into account many factors. However, there are drawbacks with using timestamps as addressed above, and a solution to this would be to use time denominated in ticks of actual activity (such as views of the 'new' page, or even upvotes-to-all-submissions) which might address this issue. The use of penalty is an interesting concept that would make sense in a user generated content system in which there are no administrators to regulate the content.

2.2.3 Wilson Confidence Sort

The problem that the Wilson confidence score tried to resolve was, developers who need their application's users to sort things. In order to balance the proportion of positive ratings with the uncertainty of a small number of observations. The math to solve this problem was worked out in 1927 by Edwin B. Wilson(Miller, Feb. 2009). Considering only positive and negative ratings (i.e. not a 5-star scale), the lower bound on the proportion of positive ratings is given by:

$$\left(\hat{p} + \frac{z_{\alpha/2}^{2}}{2n} \pm z_{\alpha/2}\sqrt{[\hat{p}(1-\hat{p}) + z_{\alpha/2}^{2}/4n]/n}\right)/(1 + z_{\alpha/2}^{2}/n).$$

Figure 2.3: Edwin B. Wilson confidence sort in mathematical notation

2.2.3.1 Conclusion

Despite a very sought after solution to modern-day sorting, the Wilson Confidence score is to some degree. This very complicated solution only increases the efficiency of the results by a small margin. Furthermore, many problems occur in the voting system that the Wilson Confidence sort is based on. A simple example is that many people who find something mediocre will not bother to rate it at all; viewing or purchasing something and declining to rate; it contains useful information about that item's quality (Miller, Feb. 2009).

2.2.4 Summary

To summarise, most of these algorithms mentioned are tailored for their particular use or website. However, most of the existing solutions are bound by the user's interaction with the content to improve or worsen the overall score of a post. What this project is trying to propose is a system where the only interaction that is needed by the reader is their interests, after which the algorithm takes care of the rankings. Nonetheless, we can assimilate parts of existing systems that can help polish the final version of the algorithm.

Chapter 3

The Algorithm

3.1 Planning

Most ranking algorithms as have been discussed, mainly revolve around some combination of voting and time. While some have more complicated parameters, they are usually tailored to the website's philosophy, such as the use of the penaltiy attribute in Hacker Rank. However, since this project aims to provide the user with an automated display of the most relevant news tailored to the user's interests, voting will be replaced with the user's interests.

The user should be able to select interests in their account page, which have a score associated with them and be a significant contributor in the score of a post. When a post gets created is also be a valuable attribute, and it should be beneficial to the algorithm since time can never be the same even if the rest of the attributes are the same, differentiating in that way two potentially similar posts. Another vital aspect to consider is how many interests does the post have, in which the user is not interested. However, the impact of this attribute should not be as negatively equal to a relevant interest in a post. For example, if the score of the common interest between user and post is equal to 10, a non-common interest should impact the overall score by -5 or lower. Lastly, the concept of banning or negatively reducing a ranking score is an essential attribute and can be detrimental in the image and environment that a system portrays.

$$score = t_u ,$$

where t_u is the timestamp that the post was updated.

$$score = \left(\sum_{i=0}^{n-1} a_i * 5 \right) * score ,$$

where a_i are the common interests of the post and the user.

$$score = \frac{score}{\left(\sum_{i=0}^{n-1} b_i * 2 \right)} ,$$

where b_i are the non - common interests of the post and the user.

$$score = \frac{score}{\left(\sum_{i=0}^{n-1} c_i * 3 \right)} ,$$

where c_i are the number of unique penalty words in the post's content.

Figure 3.1: The stages of the final ranking algorithm.

3.2 Designing

After a couple of failed attempts, the final algorithm was a combination time, common interests, non-common interests and penalties. The algorithm relies on multiplication and division to increase or decrease a rating for the sake of simplicity bundled with the fact that time which when denoted in timestamps usually tends to be a couple of digits long in the first place.

Since time was going to be used as timestamps which usually tend to be a couple of digits long, the thought of using multiplication and division to increase or decrease a rating was used. As portrayed in 3.1, if the user is not logged in or, they have no interest selected, the score of each post is set equal to the time that the post was updated. However, if the user is logged in and has selected interests, then the existing score is multiplied by 5 for each common interest of the user and the post. Then for the deduction part of the algorithm, the score is halved for each non-common interest. This decision makes sense since if a potential user is interested in news, a post with only the "news" interest will rank higher than a post with the news and music interest unless the first post is much older than the second one. The last deduction part of the algorithm is penalties. Penalties that at the moment are profanity words, essentially counter a common interest, since the score of a post is divided by 5 for each penalized word the post contains.

3.3 Creating

Assigning each post with a score based on the parameters aforementioned happens when the "get_blog_queryset" function is called in "home_screen_view".

```
# Checks if the user is authenticated.
if user.is_authenticated:
for q in queryset:

  # A list of common interests.
  common_result = []
  # A list of non-common interests.
  non_common_result = []
  # Set's the score to the time a post is updated.
  q.post_score = q.date_updated.timestamp()

  for element in q.interests:
    # Updates based on common interests.
    if element in user.interests:
      common_result.append(element)
      q.post_score = q.post_score * (len(common_result) * 5)

    # Updates based on common interests.
    if element not in user.interests:
      non_common_result.append(element)
      q.post_score = q.post_score / (len(non_common_result) * 2)

    # Updates based on non-common interests.
    if q.title in profanity_list or q.body in profanity_list:
      for penalty in penalty_list:
        q.post_score = q.post_score / (len(penalty_list) * 5)
```

Chapter 4

System Analysis

4.1 Analysis of Existing Systems

4.2 Feasibility and System Limitations

4.3 User Experience

4.4 Proposed Solutions

4.5 Interview & Survey

Chapter 5

System Design

5.1 Prototype

5.2 Design Overview

5.3 User Interface Design and Usability

5.4 Difficulties faced

5.5 Main Implementation

Chapter 6

Testing & Evaluation

Chapter 7

Statement of Ethics

Chapter 8

Conclusion