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

© Copyright 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

A_f	The source message, being a sequence of f source symbols $a_1 a_2 \dots a_f$
a_i	The i^{th} symbol in the source message, where $a_i \in S_m$
B_g	The decoded message, being a sequence of g source symbols $b_1 b_2 \dots b_g$
b_i	The i^{th} symbol in the decoded message, where $b_i \in S_m$
C_h	The transmitted (compressed) message, being a sequence of h Tunstall codewords $c_1 c_2 \dots c_h$
c_i	The i^{th} codeword in the transmitted message, where $c_i \in T_n$
D_h	The received (compressed) message, which for a complete Tunstall code is a sequence of h Tunstall codewords $d_1 d_2 \dots d_h$
d_i	The i^{th} codeword in the received message, where $d_i \in T_n$ for a complete Tunstall code

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 Design Process and Technologies used

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

1.3.1 Django Framework

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. Furthermore, Django is based on the python programming as the main language and HTML for the front-end and SQLite as a relational database. Moreover, Django uses an MVC pattern similar to the more widely used MVC that frameworks like Ruby and Ruby on Rails use.

1.3.2 Software Development Lifecycle

1.3.3 Chosen Methodology

1.3.4 Report Structure

Chapter 2

Literature Review

2.1 Reddit Rank (Hot Sort)

After conducting my research, it is 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.

2.1.1 Weaknesses

1. 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).

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 $f(t_s, y, z)$

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

Figure 2.1: Reddit Ranking Algorithm in mathematical notation

2.1.2 Conclusion

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 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

2.2.1 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 new submissions are getting.

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 Frameworks

2.2.4 Conclusion

Chapter 3

Constructing the Algorithm

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