Bursting Filter Bubbles With Programmed Serendipity

Master's Thesis

Alexander Simes

Bursting Filter Bubbles With Programmed Serendipity

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE
TRACK SOFTWARE TECHNOLOGY

by

Alexander Simes
born in Paris

Web Information Systems
Department of Software Technology
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
http://wis.ewi.tudelft.nl

Sanoma
Capellalaan 65, 2132 JL
Hoofddorp, The Netherlands
http://www.sanoma.nl/

Bursting Filter Bubbles With Programmed Serendipity

Author: Alexander Simes
Student id: 4415299
Email: alex.simes29@gmail.com

Abstract

When talking about personalization online, Google CEO Eric Schmidt recently said "it will be very hard for people to watch or consume something that has not in some sense been tailored for them." This level of personalized filtering of content has worried academics and activists. Many argue that users will be trapped in a so-called "Filter Bubble," limiting their exposure to challenging or new ideas they are not expected to like. In answer to these worries, serendipitous recommendation systems have been developed to help users make surprising and pleasant discoveries outside of their typical online content sphere. In this thesis, we investigate whether serendipitous recommendation system do indeed help break the filter bubble effect. In contrast to previous work, we investigate this question on one large-scale live experiment. We find that serendipity partially mitigates the filter bubble effect, but that users are more responsible for their own filter bubbles than algorithms. Further, we search for user characteristics that can be used to identify users more likely to experience the filter bubble effect. We find that users spending more time on site are less likely to experience the filter bubble effect.

Thesis Committee:

Chair:	Prof. Dr. Ir. G.J.P.M. Houben, Faculty EEMCS, TU Delft
University supervisor:	Dr. C. Hauff, Faculty EEMCS, TU Delft
Company supervisor:	S. Kieft, Sanoma Media Netherlands, Hoofddorp
Committee Member:	Dr. Z. Al-Ars, Faculty EEMCS, TU Delft

Preface

When I reached the point in my studies to decide what to focus on in my thesis, I was unsure what I wanted to pursue at first. The debate around the filter bubble effect caught my interest because of its ethical implications and impact on our ever-advancing technological society. Investigating a topic anyone using the internet could easily relate to also appealed to me. Further, being able to build my experiment into a live platform added the software development challenge I was looking for in my studies here. I was fortunate to have advisers at the TU Delft and Sanoma who fully supported my research direction.

I would like to first and foremost thank my adviser Claudia Hauff for always providing excellent feedback and guidance over the last nine months. Thanks to Claudia's expertise, I have gone from a student uncomfortable with academic writing to confidently submitting this thesis. Of course, I spent most of my time working at Sanoma in the company of the big data team and content platform team. Having a productive work environment with intelligent coworkers helped me to complete this work in a timely manner. Daily standups and lunches were always something to look forward to in the work day. Additionally, I owe a big thank you to Jelmer Voogel for bouncing ideas back and forth and helping me develop my vision into a reality on the Sanoma platform.

Of course, I would be remiss not to mention the unfailing support from my family, my girlfriend McKenna, and my friends. Having people outside of work life taking an interest in what I was studying and my progress meant a lot.

Now, finally, I present my masters thesis to you. I hope you find the filter bubble effect as interesting as I do and enjoy my take on it!

Alexander Simes
Delft, the Netherlands
October 13, 2016

Contents

List of Figures

Chapter 1

Introduction

With the mountain of content at our fingertips on the web, personalization helps us find what most appeals to our interests. Examples of such personalization include the *Recommended for You* box on Google News, or the *Discover Weekly*[1] playlist produced by Spotify. However, there are also cases of personalization algorithms acting in the background without notifying the user of personalized modifications to their web experience.

Consider Google Search, where results are discreetly modified according to a wide range of signals; including location, language, and browsing history [22, 45]. Figures 1.1 and 1.2 show an example of how Google Image search results are modified according to expected user preferences. We use the search term *crimea russia* in English and Russian using a VPN to America and Russia respectively. The search results differ significantly to appeal to the estimated user interests; Americans see images about conflict and invasion while Russians see images of patriotism and pro-annexation. Another example is Facebook's filtering of users' Newsfeeds according to a host of user-specific variables, favoring items users are more likely to click on [32]. Examples of filtering through personalization can be found across countless domains; news, music, movies, social media streams, and many more. In short, personalization algorithms are constantly modifying users' online experiences to help them find content expected to be more useful to them.

The prevalence of personalization algorithms has worried academics and web activists [45, 55, 62]. They argue that personalization separates users into individual, algorithmically defined bubbles of content. Furthering concern, content favored by typical personalization algorithms is popular, obvious, or already known to the user [25, 65]. Eli Pariser has described these worries in his book about the dangers of personalization, titled *The Filter Bubble: What the Internet Is Hiding from You* [45]. He calls content favored by typical personalization algorithms *information junk food*, since it is easily and eagerly consumed by users. In contrast, content that is new to a user or challenges his or her point of view is dubbed as *information vegetables*. Pariser argues that as users consume information junk food, personalization algorithms favor information junk food more and more. In turn, this exposes users to continually less information vegetables, making them even less likely to consume them. Pariser coined the term *Filter Bubble* to describe this effect, which he defines as "a self-reinforcing

[1] https://www.spotify.com/us/discoverweekly/

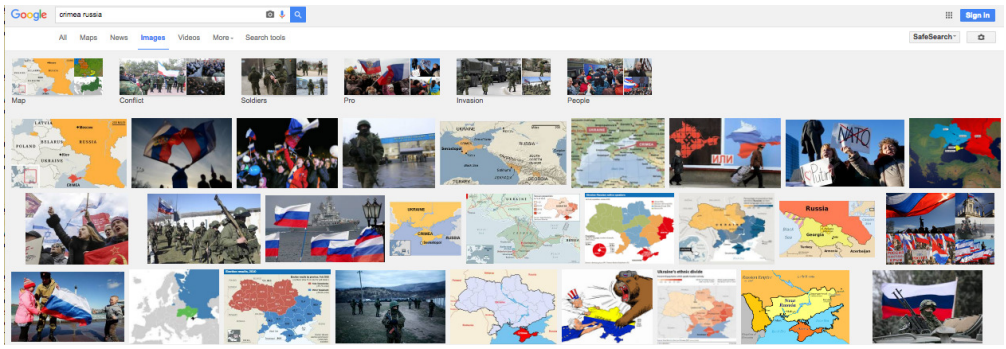


Figure 1.1: Google Image results for *crimea russia* from a location in the United States. Images of soldiers and are suggested too add "conflict" or "invasion" to our search. *Screenshot taken May 23, 2016*

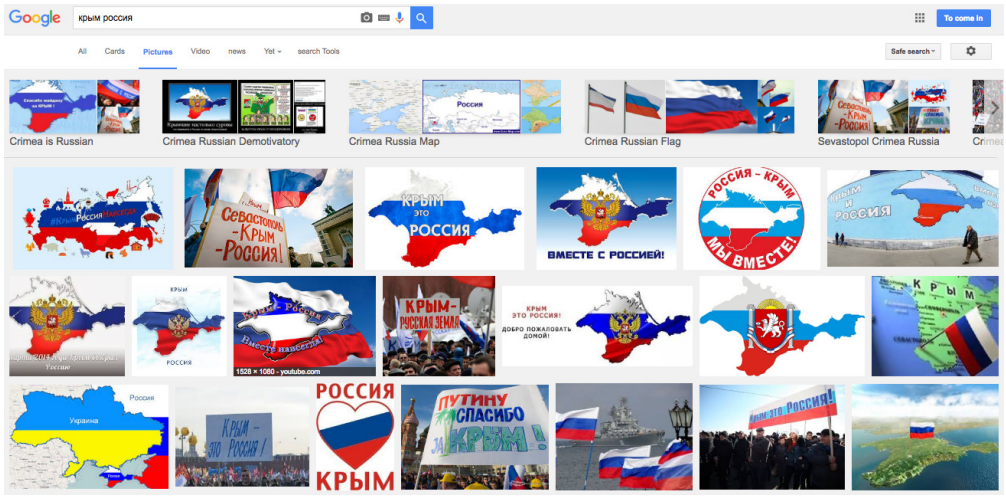


Figure 1.2: Google Image results for *crimea russia* in Russian from a location in Russia. Pro-russia imagery and patriotism are given higher ranking. *Screenshot taken May 23, 2016*

pattern of narrowing exposure that reduces user creativity, learning, and connection."

Eli Pariser's book [45] has become a cornerstone in the ongoing debate about the negative effects of personalization on the internet and whether or not the filter bubble exists. Since its publication, Pariser's book has been cited over 1,400 times, showing that it has made an impact in the academic community. Most of the citing research only mention the filter bubble as a problem caused by personalization [14, 17, 65, 33, 56], but some attempt to prove or disprove its existence [38, 23, 6, 40, 39]. Facebook has recently released a study to put the filter bubble theory to the test [6], finding that Facebook users' content is slightly skewed to match personal preferences and beliefs. However, they observed individual user choices caused more of a filter bubble than algorithmic filtering, indicating that the filter bubble effect they observed is self-imposed. Nikolov et al. [39] compare the prevalence of the filter bubble effect in online information search, email exchanges, and communication. They find more of

a filter bubble effect in traffic from social media than search engine and email traffic. Another study found YouTube users experienced a filter bubble in the political leanings of the videos they were exposed to [40]. Other researchers have had different results however, showing little or no evidence of a filter bubble in their experiments [38, 23]. The filter bubble effect debate is ongoing in the academic community.

There has also been much discussion outside academic publications. The popular technology magazine *Wired* has published a number of articles discussing both sides of the filter bubble debate[2,3,4]. The *New York Times* published an article siding with Pariser [53] and hosted a debate on the matter[5]. Some argue that it is up to individual users to fight the filter bubble by going out of their way to consume information vegetables[6], rather than expecting algorithms to do it for us. In contrast, the primary inventor of the Amazon item-item recommendation engine argues that personalized recommenders help users discover items they otherwise would not find, rather than constrain users[7]. A journalist and editor at *Slate* also discounts the filter bubble theory, providing examples of users from different backgrounds having similar results across many Google searches [61]. As in the academic comunity, discussions in the non-academic community both support and criticize the filter bubble effect.

A specific aspect of personalization where the filter bubble effect is hotly debated is recommender systems, which is what we focus on in this thesis. Recommender systems are software tools and techniques that provide content suggestions to users [52]. Several researchers [33, 14, 56, 65, 17] argue that recommender systems actually mitigate the filter bubble effect by helping users discover new content through serendipitous[8] discovery. A user makes a serendipitous discovery through a recommender system occurs when they are suggested a surprisingly interesting item they may not have found by themselves [21]. Using Pariser's terminology, potentially serendipitous recommendations can be seen as information vegetables. Researchers developing serendipitous recommendation systems believe they can provide users with information vegetables they would not otherwise find. This would mean specialized serendipitous personalization algorithms could, in fact, reverse the filter bubble effect.

While we have found much research regarding the design of serendipitous recommenders, there is a conspicuous lack of research **measuring** how serendipity in a recommender system changes the filter bubble effect. Serendipitous recommenders are typically evaluated according to how unexpected, surprising, and useful recommendations are. In contrast, the filter bubble effect is measured in considering the diversity of content consumed by and recommended to the user changes over time. There has been some work to measure the filter bubble effect created by non-serendipitous recommenders [38, 23], but those studies are not on live datasets. By *live dataset*, we mean a study using real users on a real platform who are not aware of the experiment.

[2]http://www.wired.com/2012/01/google-filter-bubble/

[3]http://www.wired.com/2014/08/i-liked-everything-i-saw-on-facebook-for-two-days-heres-what-it-did-to-me/

[4]http://www.wired.com/2011/03/eli-pariser-at-ted/

[5]http://www.nytimes.com/roomfordebate/2012/10/22/reading-more-but-learning-less

[6]https://onlinejournalismblog.com/2016/06/28/dont-blame-facebook-for-your-own-filter-bubble/

[7]http://glinden.blogspot.nl/2011/05/eli-pariser-is-wrong.html

[8]Serendipity means a "fortunate happenstance" or "pleasant surprise". The word originates from the tale of *The Three Princes of Serendip*, who were always making accidental discoveries of things which they were not in quest of"

Additionally, we have found little work proposing possible causes of the filter bubble effect other than personalization algorithms. There is minimal previous work investigating user characteristics that help identify users experiencing the filter bubble effect. We aim to fill these gaps in our research.

In this thesis, we investigate two questions about recommender system's role in the filter bubble effect. First, we explore the relationship of serendipity in recommender systems and the filter bubble effect at the level of individual users. Second, we consider the possibility that personalization algorithms are not solely responsible for the filter bubble effect, and investigate whether certain user characteristics are tied to the filter bubble effect. We perform an experiment with over 9,000 users on a live discussion forum website using an established recommender system and a specialized serendipitous recommender. Using the datasets generated in our experiment, we analyze the effect of the type of recommender on the filter bubble effect and whether certain user characteristics indicate more of a tendency towards a filter bubble.

1.1 Research Objectives

Both of our research questions are concerned with the filter bubble. Specifically, we are interested to see how different factors mitigate the filter bubble effect. A *mitigation of the filter bubble effect* means that individual users are experiencing less of a filter bubble effect in their experience online. Our first research question is focused on the relationship between serendipity in a recommender systems on the filter bubble effect.

RQ1: How does serendipity in a recommender systems affect mitigation of the filter bubble effect?

We are also concerned with other possible factors influencing the filter bubble effect. Our second research question is aimed at exploring factors besides the type of recommendation causing a filter bubble effect. The user characteristics we chose to compare against individual filter bubbles are: click-through rate on recommendations, initial diversity of user browsing history, and user activity levels. Click-through rate (CTR) is a measure of how often users click on a recommendation when it is shown to them.

RQ2: What user characteristics affect mitigation of the filter bubble effect?

1.2 Use Case: Sanoma

The experiments and research for this thesis was performed at Sanoma Media Netherlands[9]. The Sanoma Media Netherlands group is the largest media company in the Netherlands, and has operations in 12 other countries. In addition to publishing magazines and television, Sanoma publishes a range of websites including across a variety of domains including news[10], car trading[11], online shopping[12], and fashion[13]. We

[9] www.sanoma.nl
[10] www.nu.nl
[11] www.autoweek.nl
[12] www.kieskeurig.nl
[13] www.fashionchick.nl

ran our experiments on the Viva Forum website[14]. Viva Forum is a Dutch discussion site targeted primarily at women. We chose this site after careful consideration of the option available, as is detailed in Section 3.1.

1.3 Contributions

This thesis has a number of contributions, which we itemize below.

- We investigate the impact of personalized recommender systems on the filter bubble effect in a new domain, backed by live data.
- Ours is the first study to our knowledge to compare measure the effect of serendipitous recommenders versus standard recommenders on the filter bubble effect.
- We investigate the connection between certain measurable user characteristics and presence of filter bubbles.
- We open source code[15] of a scalable, graph-based serendipitous recommender that was deployed in a production environment.

While our primary focus is furthering research in the filter bubble debate, we also hope that our open source serendipitous recommender code will aid academics in the serendipitous recommender systems domain.

1.4 Thesis Outline

In Chapter 2, we discuss the recent history and state-of-the-art in research regarding the filter bubble effect and serendipitous recommendation systems. An overview of our approach, including a description of the chosen content domain and recommender algorithms, is given in Chapter 3. We detail our implementation and experiments in Chapter 4, followed by a description and discussion of our results in Chapter 5. Finally, we conclude our work in Chapter 6.

[14] forum.viva.nl

[15] github.com/alex9311/predictionio-serendipitous-recommender-template

Chapter 2

Related Work

In this chapter we cover the background work in the fields touched upon in this thesis. We detail research on the filter bubble effect and related subjects in Sections 2.1 and 2.2. Serendipity in recommender systems is covered in Section 2.3.

2.1 Precursors To Filter Bubble Studies

The impact of recommender systems on user experience has been studied over the years. Senecal et al. [50] found users to be more influenced by online recommendations than by human recommendations. In 2006, Amazon reported that 35% of its sales came from its recommendation systems [35], revealing recommenders to be a valuable part of generating revenue. Similarly, in 2012, Netflix reported that users found 75% of their content through the recommendation system [4]. Besides statistics on recommender usage, there have been studies examining different effects of recommenders on user consumption. Despite the popularity of Pariser's viewpoint [45] however, there has been little effort to measure the filter bubble effect specifically. We speculate that the lack of research is because there is little economic motivation to study the filter bubble, since companies are most concerned with generating revenue and retaining users rather than worrying about putting their users in filter bubbles. However, there is a set of older concepts related to the filter bubble which have been given more attention in research: the **long tail effect**, **content diversity**, and **user fragmentation**.

The first studies to investigate the effect of recommender systems on user consumption had economic motivations. In 2006, Anderson et al. [5] argued that, since online stores stock much more content than brick-and-mortar stores, niche products can collectively create more revenue than popular products. Niche products are those which appeal to a small subset of users. This retailing strategy of pushing users towards niche products is called the *Long Tail*. A classic example of Long Tail retailing, told by *Wired*, is from the Amazon online bookstore[1]. In 1988, a hiking survival story, *Touching the Void*, was published and had modest success. Eight years later, a book with a similar premise called *Into Thin Air* was published and sold many more copies. Soon after the publication of the second book, *Touching the Void* started selling much more, eventually surpassing *Into Thin Air*. This would not have been possible in a brick and mortar store, especially considering that *Touching the Void* was nearly out of

[1] http://www.wired.com/2004/10/tail/

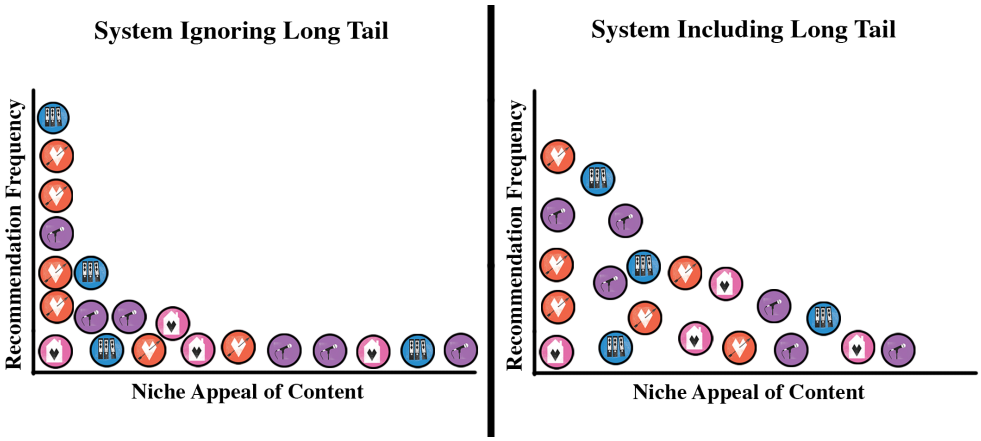


Figure 2.1: Systems without and with long tail retailing. Each circle represents a distinct item, with each icon representing a category of items. The difference can be seen in the recommendation frequency of items with higher niche-appeal.

print by 1996. The Amazon recommender system brought a then niche product to the correct users' attention, causing its sales to increase.

Figure 2.1 shows the difference between a recommender that is targeting long tail items versus one that is not. What researchers have since investigated is whether recommenders are indeed pushing users towards niche products (right side of the figure) or simply towards popular products that appeal to most users (left side of the figure). Tucker et al. [58] studied the long tail on an online vendor directory, where both popularity-based and alphabetical ranking was used. They found that users on the popularity-based system were being pushed towards universally popular items, but that niche product sales were not decreasing. Another study [15] confirmed that recommenders push users towards common popular items, but noted that users were still discovering new products with the recommender. In contrast, Oestreicher et al. [41] found that Amazon product categories influenced by recommenders to have a much flatter demand curve than other products. In other words, the recommender was equalizing sales across products rather than increasing sales in the most popular items. Park et al. [46] and Yin et al. [64] take a different route and successfully design recommenders specialized to help realize the Long Tail retailing strategy. While researches have successfully built long-tail targeting systems, it is unclear if the Long Tail retailing strategy is intrinsically present in online environments.

Long tail is related to the filter bubble effect because both are concerned with how an online environment is shaping user behavior. However, long tail studies differ from filter bubbles in a few key ways. First, the filter bubble effect is concerned with individual user experiences, while long tail studies investigate overall content consumption patterns. Product designers trying to engineer systems to make more profit from long tail items are concerned with recommending less popular items that appeal more to specific users. In contrast, those designing systems to counteract the filter bubble effect are more interested in users discovering new content outside of their bubble, regardless of content popularity.

Aside from long tail studies, another precursor to filter bubble research are studies

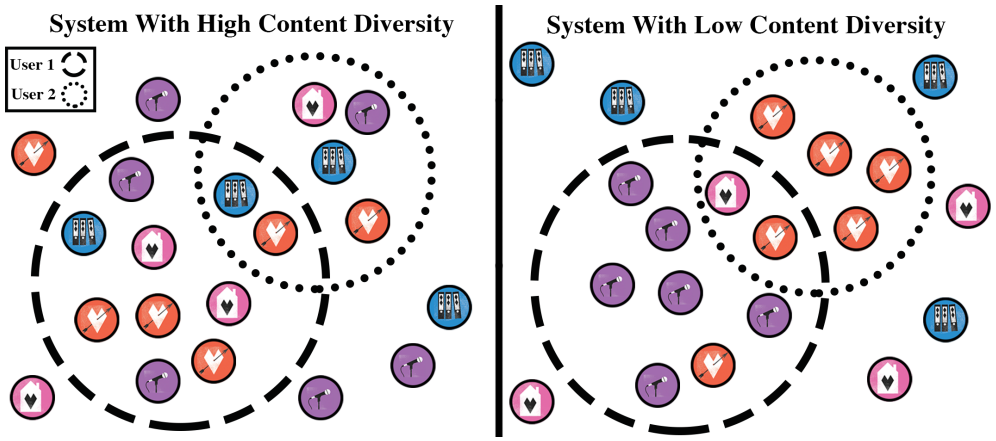


Figure 2.2: Systems with and without content diversity, circles represent the content individual users are exposed to.

looking at *content diversity* and its role in user satisfaction. As stated in [52], users will more likely find a suitable item if there is some degree of diversity among recommended items. Ratner et al. [47] found that users sometimes choose less-preferred items for the sake of variety, showing a desire for some diversity. Diversity in recommenders can be broken up into two camps, individual diversity and aggregate diversity. Individual diversity is how varied individual's consumption and recommendations are, while aggregate diversity considers the diversity across all users [3]. In Figure 2.2, the right side has slightly lower aggregate diversity but much lower individual diversity than the left side. The opposite is also possible, when aggregate diversity is low with high individual diversity [15]. For example, if a system is recommending the same five items to all users, the five items might be a diverse set of content, but the aggregate diversity will still be low. Fleder et al. [16] showed that recommenders can cause overall diversity to decrease while individual diversity increases. Brynjolfsson et al. [11] showed that the online environment, including recommenders, causes sales diversity to increase compared to brick-and-mortar stores. Research and findings in diversity studies are quite close to the long tail studies, with the key difference being that diversity studies do not consider item popularity.

An important note on diversity is the difference between studying the diversity of content recommended versus content consumed. The two notions are not necessarily the same. A user could be recommended a very diverse list of items; yet always click on items from the same narrow window of content. Further, consider a user who is interested in discussions about cars and discussions about marriage. If this user is shown one recommendation set with car and marriage discussions and another with discussions about boats, the former is more diverse but the latter will surely help the user break out of their filter bubble more. This is where the difference between diversity studies and the filter bubble effect lies. With the filter bubble, we are concerned with both the diversity of content consumed and content recommended, and how the two change together over time. Further, comparing the user history to recommended content is important to measuring the filter bubble effect.

Another topic close to the filter bubble effect is *user fragmentation*. User frag-

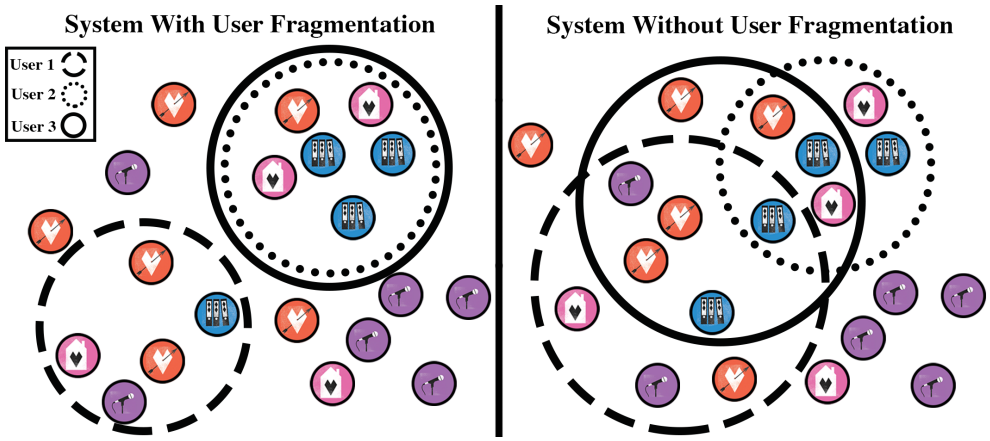


Figure 2.3: Systems with and without user fragmentation, circles represent the content individual users are exposed to.

mentation is the concept of users across a system having different experiences due to personalized recommendation and filtering, causing the community to become fragmented. Web activists and academics are concerned that fragmentation negatively affects social discourse since people have a narrow information base with little in common with one another [45, 55, 57]. User fragmentation is measured by the dissimilarity of user consumption across a system, as is shown in Figure 2.3. In the left panel, users have been broken into two tribes of experience. In the right panel, the user experiences intermingle, indicating lower fragmentation.

There have been many studies on user fragmentation in online environments across many platforms. Gilbert et al. [19] and Adamic et al. [2] investigated blogs and found clearly fragmented communities, where users interact far more with users of similar opinions than those with contrasting opinions. Similar findings have been published for Twitter [13] and Facebook [6]. However, these studies do not consider the role of the recommender system specifically. In contrast to the many arguments asserting that recommender systems fragment users, Hosanagar et al. [23] found that recommender systems cause a decrease in fragmentation between users. More research needs to be done to confirm how recommenders systems play into user fragmentation, but there is consensus on online environments in general.

User fragmentation is certainly tied to filter bubbling but it is not entirely the same. Both problems are concerned with how recommenders limit users' exposure to a specific subset of content. However, users could have a very narrow exposure but, if exposure is identical across all users, there would be no fragmentation. In other words, the filter bubble effect is concerned with individual users, while fragmentation is concerned with the community as a whole.

We have shown how diversity, long tail, and user fragmentation differ from the filter bubble effect. Despite these differences, the concepts are all undeniably intertwined. In studies since the filter bubble was defined [45], researchers have taken this past work as groundwork and modified it to study the filter bubble effect.

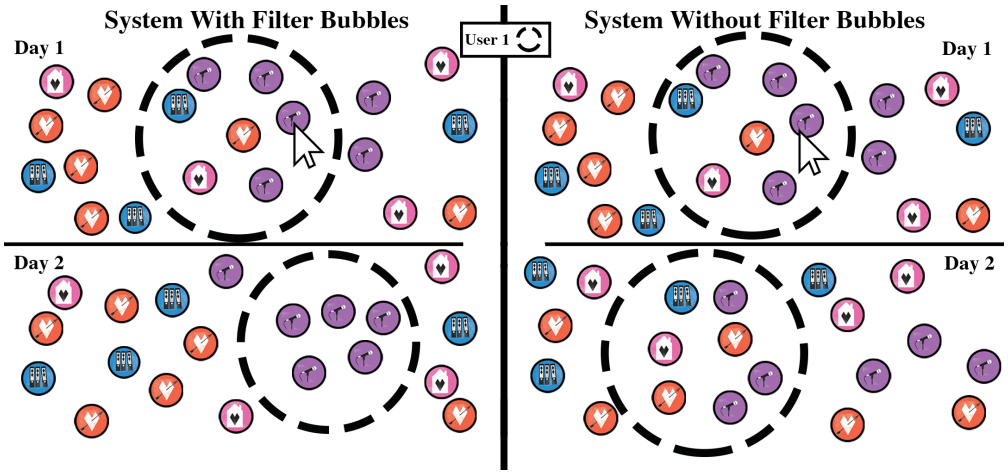


Figure 2.4: Systems with and without filter bubble effect. The content the user is exposed to is shown in the dashed circles. The scope and diversity of content the user is exposed to is narrowed from day 1 to day 2 when there is a filter bubble effect. This does not occur in the scenario without a filter bubble ffect.

2.2 Filter Bubble

Let us recall Eli Pariser's definition of a *Filter Bubble*, "**a self-reinforcing pattern of narrowing exposure that reduces user creativity, learning, and connection.**" We identify two key aspects to Pariser's definition: *self-reinforcing* and *narrowing*. First, we interpret *self-reinforcing* to mean that both the user and personalization algorithm play a part in the filter bubble effect. The user is selecting a narrowing set of items, which, in turn, leads the personalization algorithm to serve a narrowing set of items, and so on and so forth. Considering both the behavior of both users and the personalization algorithm as parts of the filter bubble effect has been done in previous work as well [6, 38]. Second, we interpret *narrowing* to mean that the effect worsens over time. Considering trends over time to study the filter bubble effect has also been done previously [38]. With this definition of filter bubbles in mind, we will examine the existing work that focuses on the filter bubble effect.

Figure 2.4 shows a visual representation of the filter bubble effect. On day one in the left panel, the user's recommendations have some preference towards entertainment (purple) discussions. Since the user then clicks more on entertainment discussions, the system narrows its scope of recommended content on day two. In contrast, the right panel shows a system which always exposes the users to content outside of their bubble of normal consumption.

Filter bubble specific studies began with Eli Pariser's defining of the concept in 2011 [45]. Since then, there has been work investigating ways of breaking users out of their filter bubbles. Bozdag et al. [10] and Nagulendra et al. [37] develop applications to help users understand their own filter bubble. The hope is that users will actively make an effort to change their consumption patterns if they are made aware of their own biases. Bozdag et al. [10] do not evaluate their tool on users but Nagulendra et al. [37] found that the visualization leads to increased user awareness of the filter

bubble, understandability of the filtering algorithm, and to a feeling of control over their online experience. Similarly, Graells et al. [20] research how best to present users with political content contrasting with their own viewpoints. The problem with these types of studies assume the filter bubble exists and do not seek to prove or disprove the existence of the filter bubble effect.

Many recommender systems research publications have mentioned the filter bubble effect in their work [23, 65, 33, 56, 14, 38, 17]. The problem we have found is that most scholars reference the filter bubble effect as a motivation for their work in improving recommender systems, yet do not attempt to measure how their improvements alleviate the problem. For example, in [14, 17] develop serendipitous recommenders system to counteract the filter bubble effect. However, only the serendipity of the recommender is evaluated, rather than its possible mitigation of the filter bubble effect.

Taramigkou et al. [56] present a recommendation system to allow users to consciously break out of their filter bubbles. Their system allows users to explicitly choose to explore a new type of music outside of their normal consumption pattern. Once the user selects a target genre, the application guides them through a transition from familiar content to the new content. While users generally enjoyed the experience and discovered new music in the qualitative evaluation, this experiment does not show whether or not the filter bubble effect actually occurs. Instead of studying a system to see if a filter bubble effect exists, they provide users with a way of actively discovering self-selected new content.

Similarly, Knijnenburg et al. [29] recently developed recommendation systems designed to help users in developing, exploring, and understanding their own unique tastes and preferences. They call these recommendation systems *Recommender Systems of Self-Actualization* (RSSA). RSSA prioritize items other than typical Top-N systems. Four RSSA prioritization ideas are presented: "Things we think you will hate", "Things we have no clue about", "Things you'll be among the first to try", and "Things that are polarizing." Knijnenburg et al's work [29] is a proposal for a new path in research and does not provide evaluations for their systems.

Other researchers have attempted to measure the filter bubble effect. Nikolov et al. [39] study the filter bubble effect across different online medium. The mediums they examine are information search, communication from email exchanges, and communication captured from social media streams. Their dataset spans 41 months and comprises 106 million clicks on links from a search engine, email client, or social media site, that target one of almost 7.18 million distinct content items. They find that the diversity of targets reached from social media is significantly lower than those reached from search engine traffic. Further, the differences in diversity did not change significantly over the 41 months of data. Email was also found to bring users to a less diverse set of links than search, but of weaker statistical significance than social media. The downside of this study is that the analysis is not done at the level of individual users, and the presence of a collective filter bubble does not imply the presence of individual filter bubbles. Additionally, Nikolov et al. [39] do not study the impact of recommender systems, only the online medium itself.

Baskshy et al. [6] investigated the filter bubble effect on Facebook in the context of political content. They use a comprehensive dataset from Facebook from a 6-month period. The data includes just over 10 million users and 226,000 distinct content URLs. To analyze the filter bubble effect, they compare the subset of content that is shown to

users on their feed with the broad set of available content. They also observe which information users decide to consume (click on). The key observation from their study is that the users themselves rather than algorithms limit their consumption of content outside of their bubble. While Facebook's personalization algorithms did limit some exposure to outside viewpoints, users did more to limit themselves through the content they chose to consume.

Baskshy et al.'s [6] study have some shortcomings however. A user's bubble of content is determined by the political affiliation they provide Facebook, rather than their history of consumed content. Futher, they do not discuss how the filter bubble changes over time, if at all. These shortcomings were addressed in another work by Nguyen et al. [38].

Nguyen et al. [38] took the most direct look at the filter bubble effect that we have seen, using a MovieLense dataset[2]. The MovieLense data contains data on when recommendations were served to the users, and when users rated movies. Recommendations in MovieLens are generated with a well-known collaborative filtering algorithm [48]. After some preprocessing, their dataset contained 1,405 users who provided 173,010 ratings on 10,560 distinct movies over the course of 21 months. They measure filter bubbles by examining the diversity of content recommended and the content consumed, considering both parts of the self-reinforcing cycle Pariser [45] describes. They also add the element of time by looking at the changes in both diversities over time, rather than considering a single temporal snapshot. They find that the recommender did indeed expose individual users to a narrowing set of content over time and that the diversity of content consumed by individual users slowly narrowed as well. These two trends in diversity of content would indicate that there was a filter bubble effect in between the users and the recommender system in the MovieLense data. However, they do point out that users who tended to follow recommendations experienced less narrowing of consumed content diversity. This would lead us to think that recommenders actually help users slowly break out of their bubbles of content.

There are some shortcomings in Nguyen et al.'s [38] work. First, domain constraints required them to estimate when users were following recommendations, rather than actively tracking clicks. Second, the recommender used in the experiment does not shed any light on how serendipity in recommender systems changes the filter bubble effect. Since serendipity is the characteristic argued to counteract the filter bubble, it would be interesting to investigate the difference between a traditional and serendipitous recommender.

We have covered research mentioning the filter bubble, a small subset of which actually attempts to measure it or deals specifically with recommender systems' influence. A summary of filter bubble focused papers is given in Table 2.1, to provide the reader with an accessible overview. Nguyen et al. [38] and Baskshy et al.'s [6] both found empirical evidence of the filter bubble effect when considering personalized recommendations. We have also found several researchers claiming serendipitous recommendations to be the answer to filter bubbles [65, 33, 56, 14, 17]. What is missing altogether is work investigating how serendipity in a recommendation system changes the filter bubble effect. This is one of the gaps in research we seek to address. First, we will review research in developing serendipitous recommender systems.

[2] www.movielens.org/

Table 2.1: Summary of research related to the filter bubble effect.

Citation	Year	Description
[20]	2013	Present application that makes users aware of their characterizing topics, concepts, and relations in a social network. Also nudges users to read content opposing their views through content recommendations.
[56]	2013	Present a recommendation system to allow users to consciously break out of their filter bubbles
[38]	2014	Measure how personalized recommendations effect individual users' filter bubbles
[37]	2014	Propose a visual application to make the user aware of their filter bubble and take control of it.
[10]	2015	Provide a list of tools and algorithms that designers have developed in order to fight filter bubbles
[39]	2015	study the filter bubble effect across different online settings.
[6]	2015	Measure the filter bubble effect on Facebook in the context of political content.
[29]	2016	Develop recommendation systems designed to help users in developing, exploring, and understanding their own unique tastes and preferences.

2.3 Serendipity

Past researchers have mentioned the filter bubble effect as a motivation for improving recommendation systems [65, 33, 56, 14, 17]. They want to improve recommenders so users discover new content they enjoy and do not get trapped in an algorithmically defined bubble. A common strategy of modifying recommender systems to combat the filter bubble we have seen is to optimize them for serendipity [65, 33, 56, 14, 17]. As mentioned previously, we have not found any research in how serendipitous recommenders actually affect the filter bubble. However, there have been plenty of publications regarding the development and evaluation of serendipitous recommenders.

2.3.1 Defining and Measuring Serendipity

The most commonly agreed upon definition of serendipity is from Herlocker et al. [21] in 2004. They define serendipitous recommendations as the ones *helping the user to find surprisingly interesting items she might not have discovered by herself.* There have been many interpretations of this definition and several proposed methods of evaluation. Evaluating serendipity is important to our work because we want to investigate how serendipity in a recommender system changes the filter bubble effect.

User questionnaires are one of the more common methods of evaluation. For example, Taramigkou et al. [56] provided serendipitous music recommendations then asked users "Did you find artists you wouldn't have found easily on your own and which you would like to listen to from now on?" This indicated whether or not the system was successfully providing serendipitous recommendations. The Auralist hybrid music recommender [65] also had a user evaluation component, asking if they had been introduced to new music they enjoyed. A project for serendipitous news recommendations [25] asked users if they were "positively surprised by this article" and were "glad they found it." A downside of these studies is the low number of participants involved, all three studies mentioned here had less than thirty participants. Further, questionnaires are typically quite a coarse-grained method of evaluation. Questionnaires have also been found to provide misleading information [51], people can answer untruthfully both intentionally or unintentionally.

Some studies use a combination of user evaluation and quantitative metrics to evaluate serendipity. A serendipity metric was proposed by Ge et al. [18] and has been used in several studies since [42, 43]. Their metric combines unexpectedness and relevance to define serendipity. The unexpected items (UNEXP) are defined as the set of recommendations which do not appear in a primitive prediction model [36], shown

in Equation 2.1. A primitive prediction model is a baseline recommender that is not optimized for serendipity. To solve for serendipity in Equation 2.2, each unexpected recommendation UNEXP_i is evaluated for relevance. If a recommendation is relevant, $relevance(UNEXP_i) = 1$, otherwise $relevance(UNEXP_i) = 0$. N is equal to the size of the set of unexpected recommendations, $|UNEXP|$. In [18], relevance is left to be judged by the user in a user study. Bordino et al. [9] used the same strategy to calculate unexpectedness, but leveraged crowdsourcing through the CrowdFlower platform to judge relevance.

$$UNEXP = serendipitous\ recommendation\ set \setminus baseline\ recommendation\ set \quad (2.1)$$

$$SERENDIP = \frac{\sum_{n=1}^{N} relevance(UNEXP_i)}{N} \quad (2.2)$$

There also exist purely quantitative methods of calculating serendipity. In De Gemmis et al.'s [14] study on graph-based serendipitous recommenders, a quantified combination of unexpectedness and relevance is used. Relevance is defined as whether or not the user rates the item higher than the average of all ratings provided by that user. Unexpectedness is defined with a combination of popularity (how much of the user base has seen the item) and item average rating. Note that the definition of unexpectedness changes across studies. De Gemmis et al. [14] argue that items that are less popular and less liked by users are more likely to be unexpected. Serendipity is then defined as the set of items that are both relevant and unexpected according to their definitions.

The Auralist study [65] takes a different take on programmatically measuring serendipity. They argue that serendipity represents the "unusualness" or "surprise" of recommendations, and can thus be represented as the similarity between items in a user's history and the recommended content. The similarity between items i and j is measured by finding what proportion of preferring users two items have in common, shown in Equation 2.3. Intuitively, the higher the proportion of preferring users between the two items, the greater the similarity value $PrefUserSimilarity$ will register.

$$PrefUserSimilarity(i,j) = \frac{\#\text{ users who prefer both } i \text{ and } j}{\sqrt{\#\text{ prefs in } i} \times \sqrt{\#\text{ prefs in } j}} \quad (2.3)$$

Since $PrefUserSimilarity$ gives higher values for more similar items, Zhang et al.'s [65] metric actually measures "unserendipity". Unserendipity is the inverse of serendipity. UnSerendipity is then measured as the average $PrefUserSimilarity$ value between the user's history and their recommended items, shown in Equation 2.4. Here, S is the set of all users, H_u is the items in user u's history, and $R_{u,n}$ gives the top n recommendations for user u.

$$\overline{unserendipity} = \sum_{u \in S} \frac{1}{|S||H_u|} \sum_{h \in H_u} \sum_{i \in R_{u,n}} \frac{PrefUserSimilarity(i,h)}{n} \quad (2.4)$$

Before moving on, there are two other metrics which are often discussed alongside serendipity: *novelty* and *diversity*. A novel recommendation is defined as a recommender system suggesting an item unknown to the user that they might have discov-

ered on their own [21, 59, 31]. A recommendations can be both novel and serendipitous, but this is not always the case. Suppose a user is an avid listener of Weezer and is served their latest album as a recommendation. If the user were not yet aware of the new album, the recommendation would be novel but not very serendipitous. However, if the recommendation was a much less popular band with a style similar to Weezer, the recommendation would be both novel and serendipitous. In earlier studies of serendipitous recommenders, some academics used novelty metrics as a way to evaluate serendipity [7, 54].

Diversity in recommender system evaluation is concerned with the diversity of the generated recommendations, rather than the diversity of content consumed by the user. While diversity and serendipity are often mentioned together, they are actually quite different. Suppose a music recommender serves four items in a page. If a user has an extensive history in browsing rock music, classical music, punk music, and jazz, a recommendation featuring an item from each category would be quite diverse but probably not serendipitous.

Overall, we have seen two main methods in evaluating serendipity. First, the method of combining unexpectedness with relevance. While the two sub-components are calculated in different ways, the overarching idea of representing serendipity by joining unexpectedness and relevance is the same. Second, the method of measuring the recommendation distance from the user profile. The key difference between the two approaches is whether or not relevance is taken into account.

2.3.2 Serendipitous Recommender Algorithms

Now that the definition and evaluation of serendipity is clear, we can discuss how academics have tried to program serendipity into their recommender systems. Some of the first research on programming for serendipity was in 2001 by Campos et al. [12], with the development of the `Max` recommendation agent. `Max` simulates human browsing behavior on the internet and searches for information that might interest the user yet does not lay in the user's focus. The modeled browsing process starts with a Google search of randomly chosen words from the profile in order to select pages that may spark new interests for the user. In their evaluation, only 7% of recommendations were unexpected, interesting, and hard to find for the user. The results showed that serendipity was hard to achieve but was possible.

Kawamae et al. [28, 27] use *innovator users* as sources for serendipitous recommendations. This strategy relies on the idea that among like-minded consumers, some are "innovative" consumers who know about some items well before their release and purchase these items soon after their release. Since these users are more familiar with items of interest and discover new items before others, their logs are more useful in making serendipitous recommendations. Their evaluations showed recommendations offered a high degree of recommendation serendipity.

A small branch of research in serendipitous recommenders exists in fusion-based recommenders. These systems attempt to find serendipitous content for the user by combining characteristics of two different items the user has previously rated positively. In Oku et al's work [42], a user selects two items as input, which the system uses to generate a recommendation list. These recommendations are generated by finding content between the two items and items related to either of the two. Their

evaluation showed that their methods produced more unexpected and serendipitous recommendations than the content-based and collaborative baselines.

Yet another family of serendipitous recommenders is based on graphs. Graphing strategies often generate serendipitous recommendations by recommending content outside of clusters of content familiar to the user, also known as *declustering*. Kamahara et al. [26] laid the foundation on declustering approaches. They introduce the idea of user communities based upon user's interests - allowing users to belong to various communities at the same time. A user's community partners may be members of different communities themselves. These communities removed from the user by one degree of separation are used as sources for serendipitous recommendation. They found their method to produce 15% less interesting recommendations to the user, but 40% more unexpected recommendation over a standard collaborative recommender. The results show that serendipity can be achieved with a small hit to accuracy.

Kamahara et al.'s [26] work was later built upon in [1], where user communities were again identified and used to find regions outside the user's direct community yet still close enough to be relevant. Their evaluation showed recommendations to be nearly as relevant as the baseline collaborative filter, yet more novel. Similarly, the `TANGENT` framework [44] recommends items that bridge gaps between a user's community and an outside community. Results showed that the `TANGENT` recommendations were more surprising and diverse than the baseline recommender optimized for relevance.

Similarly, the Declustering Auralist system [65] finds user's music bubbles in a graph with artists as nodes and similarity as edges. The item similarity in the graph is determined by the similarity of preferring users. In other words, songs with similar "fan bases" will be close in the graph. Items along the edges of user's musical bubbles are made to try to expand content horizons. An evaluation of the results showed significantly more serendipitous recommendations at the cost of some accuracy. They also note that their qualitative analysis showed users were often more satisfied with the serendipitous recommendations than the non-serendipitous baseline.

In 2013, a project for serendipitous recommendations in mobile apps [7] used a different technique to find items outside of users' familiar clusters. Apps are represented as nodes while the similarity between them create the weighted edges. Shortest paths between apps owned by the user are found and apps on the path not yet downloaded are recommended. Evaluation against a baseline collaborative filter showed that their recommender produced highly novel content and reduced over-personalization.

Building on other graph-based serendipitous recommenders, some academics have sought to improve the graphs by incorperating additional information into the graphs [33, 14, 60]. Maccatrozzo [33] proposes to include the link structure of the Linked Open Data (LOD) cloud[3] to find new connections between concepts, but the work is still in preliminary stages. LOD is a style of publishing structured data on the Web, allowing it to interlink with other LOD data [60]. Wang et al. [60] convert data from several external sources (Twitter, Last.fm, Lyric, Yahoo! Local) into their own LOD structure to generate serendipitous music recommendations. By using multi-hop implicit association between items in their graph, they are able to more easily generate serendipitous recommendations than a conventional content-based method. Similarly,

[3] lod-cloud.net

Table 2.2: Summary of serendipitous algorithms covered in this section.

Citation	Year	Classification	Algorithm Description
[12]	2001	Random Walks	Simulates human browsing behavior on the internet and searches for recommendations that might interest the user yet does not lay in the user's focus.
[26]	2005	Declustering	Recommends items to users from partially similar communities to their own.
[1]	2009	Declustering	Recommends items from communities outside the user's direct community yet still close enough to be relevant
[44]	2009	Declustering	Recommends items that bridge gaps between a user's community and an outside community.
[27]	2010	Innovators	Algorithm is based on the assumption that an item recently purchased by the innovator will surprise the follower more than other items.
[42]	2011	Fusion	Find serendipitous content for the user by combining characteristics of two different items the user has previously rated positively.
[33]	2012	LOD Graph	Proposes to leverage LOD to find new connections between content to generate serendipitous recommendations.
[65]	2012	Graph	Items along the edges of user's content bubbles are made to try to expand content horizons.
[7]	2013	Graph	Uses shortest path algorithm through content graph to find connections in content
[60]	2013	LOD Graph	Leverages several connected LOD sources to generate serendipitous recommendations.
[14]	2015	Graph	Take data from Wikipedia and WordNet to enrich their content graphs, allowing them to leverage less obvious connections between objects.

De Gemmis et al. [14] take data from Wikipedia and WordNet[4] to enrich their content graphs, allowing them to leverage less obvious connections between objects. WordNet is a lexical database for the English language that groups words into sets of synonyms. In their offline evaluation, their graph-based algorithm produced more serendipitous suggestions than the collaborative or content-based baselines. In a small user study, they found 69% of suggested items to be relevant, and 46% were judged as serendipitous.

We have summarized the publications discussed in this section in Table 2.2. In recent years, a trend towards graph-based solutions in serendipitous recommenders can be seen. We've discussed strategies that use external data to enhance graph-based solutions [14, 33, 60], but the other solutions look promising as well [7, 65, 26, 1, 44]. Using some sort of graph-based serendipitious recommender in our experiment would be most in-line with recent research.

[4] wordnet.princeton.edu/

Chapter 3

Approach

As we have discussed, our goal is to investigate how programmed serendipity and user characteristics influence the presence of the filter bubble effect in a live setting. Our research differs from previous research which was conducted in a lab setting, mostly ignored differences between users, and did not use serendipitous recommendations. We will investigate our research questions by running an A/B test where users are served recommendations generated by different algorithms. In this section we will describe the domain chosen for our experiment, define our metrics, and describe the recommendation algorithms we are using.

3.1 Domain: Viva Forum

Our domain has to meet certain requirements in order for it to be viable to be used in our research. We also needed to choose a domain owned by Sanoma, where we could deploy our own recommendation system and track user activities. To select our testing ground from the domains available at Sanoma, we defined five criteria:

1. Content is text-based, to allow for easier keyword vector representation.
2. Some diversity of content in domain, allowing for wider variety of recommendations and differences in user exposures.
3. A domain where users browse multiple content items per session and revisit the site.
4. If there is an existing recommender in the domain, we would like to see that users are currently using it.
5. Available access to the user characteristic data required for our second research question.

We investigated a number of domains available at Sanoma. Ecommerce sites, such as *Vtwonen.nl* or *Fashionchick.nl*, did not meet our second criteria since the sites do not carry a wide range of products. A TV guide website, *Tvgids.nl*, was also an option, but we found many of the television programs to lack textual descriptions which would cause difficulty in analyzing content. The website *WTF.nl*, providing links to strange news and stories, also had short text descriptions for many content items. Additionally, users on the site viewed an average of two pages per browsing session, which we found to be quite low for our third criteria. The domain we finally decided on is a discussion forum website called Viva Forum.

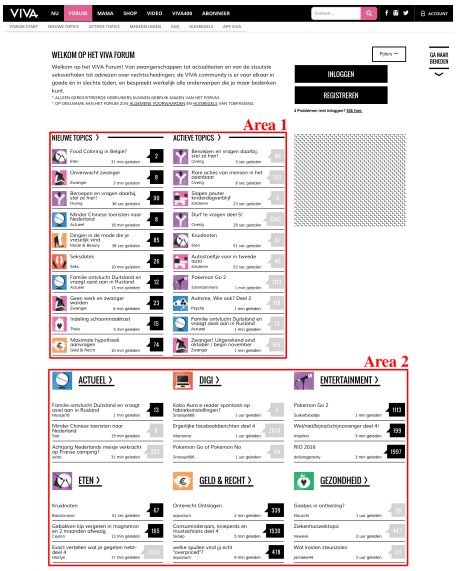


Figure 3.1: Viva home page, showing first new and active topics (Area 1) followed by discussions separated by category (Area 2).

Figure 3.2: Viva discussion page, showing original post (Area 1), the first three comments (Area 2), and the existing bottom-page advertisement (Area 3). The comments were trimmed to show the advertisement, normally users must scroll through more comments to reach the bottom.

Viva Forum is an online discussion forum targeted at women, but also has gender-neutral topics. Users can start a discussion in a selected category with questions or general remarks, after which other users can comment on the discussion with their thoughts. There are seventeen primary categories for discussion on the forum, plus eight smaller categories designated as "tip categories" or "pinboard categories." The categories range from "Relationships" and "Children" to topics such as "Money and Law" or "Travel". The top part of the Viva home page is shown in Figure 3.1. The first two columns on the home page show newly created discussions and active discussions. Active discussions are discussions which have most recently been commented on. Below the two uppermost columns, the homepage links to active discussions separated according to categories. Figure 3.2 shows the top part of a discussion page. The original post is at the top highlighted in grey, followed by the user comments.

Once we identified Viva as our most likely domain, we thoroughly evaluated it according to our five criteria. In our evaluation of the domain, we investigated one month of page view data from the Viva Foum. We limited the dataset to include only users who are tracked through the Sanoma cookie system[1], since that is how we will track users in our experiment. The data was further trimmed to include only users with at least 20 views on discussion pages over the month of time. For the first criteria, we want the content to be text-based to allow for easier representation for analysis.

[1]Sanoma uses browser cookies to track users from session to session.

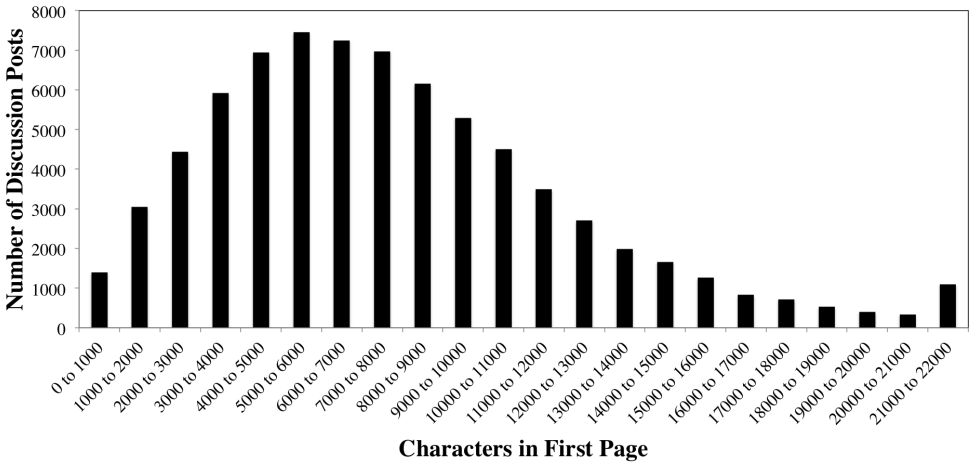


Figure 3.3: Viva content length overview for distinct items in 1 month of user view data

Viva Forums is text-based, consisting of user-generated Dutch language questions and comments. Figure 3.3 shows an overview of how many characters are in the first page of each distinct discussion post users accessed in our dataset. We made a rough calculation of average word length in the corpus and found it to be 5.4 characters, meaning 1000 characters is about 185 words. We can see that in Figure 3.3 that most discussions have well over a few thousand characters. Our first criteria is certainly met on the Viva Forum.

For the second and third criteria, the user browsing patterns and the diversity of accessed content is of interest. User *sessions* are tracked through cookies. A session is ended when the internet browser is closed, or after 30 minutes when the cookie is refreshed. We found 16,909 distinct users in our dataset with an average of 50 total sessions per user during the month of time covered. Users viewed an average of just over 5 discussion threads per session, with an average session duration of 15 minutes. As for content, there were 69,163 distinct discussions visited by users. Of the discussions visited, 2,432 were created during the time covered by our dataset, so there are approximately 80 new discussions every day. The category breakdown of the distinct items is shown in Figure 3.4. Fourteen categories have more than 1,000 items and no single category is overly-dominant, showing that content in our sample is spread across categories. We are satisfied that these statistics meet the second and third criteria.

In terms of existing recommender systems, each discussion has a four-panel advertisement at the bottom of each discussion page (Area 3 on Figure 3.2). The recommendations are either external advertisements or articles on the viva website. The four recommended items have about 310,000 views and 13,000 clicks combined, giving a little over 4% click rate on the current recommender. We deemed this to be high enough to meet our fourth criteria, as it is a reasonable click through rate in industry. However, the current recommender serves items outside of the forum rather than recommending other discussions, so we cannot use it as a baseline in our experiment. We want to serve recommendations for discussions from the forum because they offer a

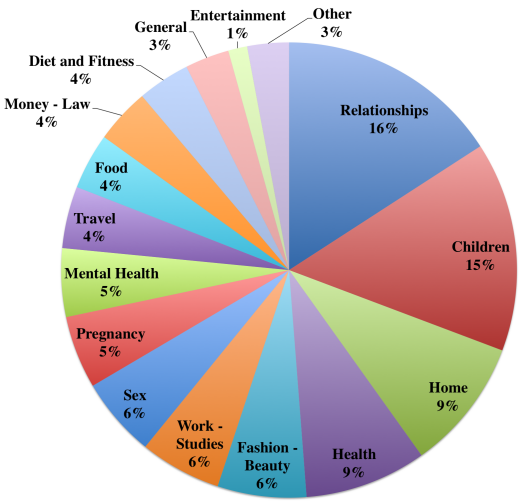


Figure 3.4: Viva content type breakdown for distinct items in 1 month of user view data

much broader range and larger pool of items than outside articles.

Finally, we checked our fifth criteria against the domain. We need to be able to measure users' initial diversity of content consumed, user click-through rate on recommendations during the study, and whether users are active. Initial diversity of content consumed is available through historical event data. Click-through rate on recommendations made during the study will be tracked with Sanoma's event tracking system. Users' level of activity on the site can be measured using log analysis on the available data. Having evaluation all of our criteria, we confidently selected Viva as our experiment domain.

3.2 Defining Metrics

3.2.1 Filter Bubble

Measuring the filter bubble is difficult. As mentioned in the Chapter 2, we break this down into two parts. First, there is a narrowing exposure of content. In our case, we are only interested in the exposure of content that comes from the recommender system. Second, there is a limited consumption of diverse content which reduces user creativity, learning, and connection.

Both parts require the measurement of the diversity across a set of content. To measure diversity in our domain, we need to know the similarity between forum discussions. One of the most popular ways of measuring similarity between text items in information retrieval is by using term frequency - inverse document frequency (tf-idf [34]). *Tf-idf* is a method of determining the importance of terms in a document. The importance of each term is increased proportionally to the number of times the term appears in the document, then offset by it's frequency across all documents. The importance of each term in a document is represented in a vector of dimension d, where d is the number of distinct terms in the entire document set.

Table 3.1: Top term weight examples from three discussions

Title: Fun things for two on vacation. Tips? **Type:** Travel	
Term	**Weight**
beach	0.4
together	0.336
city	0.191
boring	0.188
you all	0.17
vacation	0.167
week	0.16
entertain	0.149
each other	0.137
visit	0.136

Title: How long to go on vacation? **Type:** Travel	
Term	**Weight**
week	0.622
vacation	0.225
trip	0.174
mostly	0.155
midweek	0.151
year	0.151
city-trip	0.141
long	0.135
day	0.134
compromise	0.131

Title: It is over. **Type:** Relationships	
Term	**Weight**
July	0.288
marriage	0.243
suppress	0.234
intense	0.213
steal	0.176
ghost	0.173
heal	0.163
splitting	0.162
colleague	0.14
myself	0.137

To show term frequency weighting in our domain, we will consider two discussions from the Travel category of the Viva Forum and one discussion from the Relationships category. The top ten weighted terms from each discussion are show in Table 3.1. The discussion titles are shown to give an idea of its content, but the discussion content was used to generate the term vectors. We can see that the two discussions from the Travel category share some top weighted words, "week" and "vacation". In contrast, the discussion from the Relationships category does not share any top terms with the other two discussions.

Once the term vectors for each document are determined, we use *cosine similarity* to measure their proximity. Cosine similarity is an established method in informational retrieval and text mining. The forumula for cosine similarity is shown in Equation 3.1 [34], where A and B are the term vectors of two discussions to compare. With vectors that include negative values, cosine similarity ranges between -1 (exactly opposite vectors), 0 (orthogonal vectors), and 1 (exactly equal vectors). In text comparison with term frequency vectors however, the values range from 0 to 1 because term weights are not negative [34].

$$CosineSimilarity(A,B) = \frac{A \cdot B}{\|A\| \, \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{3.1}$$

To measure diversity across a set of content A, we take the average cosine distance between all items in the set. Cosine distance is $1 - CosineSimilarity$. Since similar items have similarity closest to 1, these items will have distance values closest to 0. Our equation for average diversity $AvgDiversity$ is shown below, where A is the set of items in question and V is the set of content vectors for all items.

$$AvgDiversity(A) = \frac{2}{|A|(|A|-1)} \sum_{i=1}^{|A|} \sum_{j=i+1}^{|A|-1} 1 - CosineSimilarity(V_i, V_j) \tag{3.2}$$

In addition to the two types of diversity to measure the filter bubble effect, there is the element of time. The definition of a filter bubble effect includes that it is a self-reinforcing pattern, meaning it is expected to get continually worse with time.

Including a temporal element to the evaluation is critical in accurately determining the presence of a filter bubble effect. This has been done in some previous work [38]. To this end, we will measure the filter bubble effect by considering the average diversity of content consumed by and recommended to individual users for each day in the experiment.

3.2.2 Serendipity

We covered past methods of measuring serendipity in Chapter 2, showing that there has not been an established method. Our primary metric for serendipity will follow Zhang et al.'s method [65], using distance between user history and recommendations to measure serendipity. [65] define distance from the user profile as the average distance between items in the user's history and the recommended item. They define the distance between items as a function of how similar the preferring user base is, shown in Equation 2.3. The preferring user base is the group of users who have shown preference for a given item. We follow the definition of serendipity as recommendation distance from user profile, but use a different method to find the similarity between items.

To calculate the distance between discussions, we use the tf-idf method described in the previous subsection. We chose to use tf-idf rather than Zhang et al.'s method because tf-idf is more established. Our equation for serendipity is shown in Equation 3.3, where S is the set of all users, H_u is the items in user u's history, $R_{u,n}$ gives the top n recommendations for user u, and V is the set of discussion term vectors. We use $n = 4$ because that sized recommendation panel was agreed upon with Sanoma.

$$serendipity = \sum_{u \in S} \frac{1}{|S||H_u|} \sum_{h \in H_u} \sum_{i \in R_{u,4}} \frac{1 - CosineSimilarity(V_i, V_h)}{4} \tag{3.3}$$

We do not include relevance as our formal metric for serendipity because of the challenges in measuring relevance. However, we will observe click through rate on recommendations from both the baseline and serendipitous recommender. Click through rate will give us some idea of overall relevance and whether there is a significant difference between the two recommenders in relevance.

3.2.3 User Characteristics

In addition to examining how serendipity in recommendations mitigates the filter bubble effect, our second research question regards how different user characteristics might influence the filter bubble effect. The three user characteristics we have chosen to observer are: user activity, initial user content diversity, and recommendation click through rate. We chose these three characteristics because they are measurable with the user data available to us at Sanoma. In analysis, we separate users into two groups according to these characteristics and investigate how their consumed content diversity differs. We only consider diversity of content consumed over time to measure the filter bubble effect in analysis for our second research question because the recommendations are no longer part of the issue. Further, since serendipity is not part of our second research question, we only consider users from the baseline recommendation

group. The user characteristics metrics are far more straightforward than the previous two, but we will formalize them here.

User Activity
We decided to measure user activity according to how many distinct discussions the user visited each day. Our equation for user activity across our experiment period is shown below as Equation 3.4, where *DistinctViews* is the number of distinct items viewed by user u on day d. Conceptually, *AverageDailyViewCount* shows how active a user was according to total distinct discussions viewed divided by the number of days in the experiment (29).

$$AverageDailyViewCount(u) = \frac{\sum_{d=1}^{29} DistinctViews_{(u,d)}}{29} \tag{3.4}$$

Each user's average daily view count is used to designate them as active or inactive. To separate into two groups, we take the top $\frac{2}{5}$ users by average daily view count as *active* users and the bottom $\frac{2}{5}$ as *inactive*. The middle group of users is left out of the analysis since they are average in the whole set and we are only interested in particularly inactive or active users. Nguyen et al. [38] used a similar method of leaving "average" users out of their user groups. We chose the $\frac{2}{5}$ to leave enough users in each group to have significant data.

Initial User Content Diversity
Initial user content diversity is a measure of how diverse users' consumed content set was before the start of the experiment. We consider the month of data before the start of our experiment. In this month, we calculate the daily diversity of content consumed. Diversity is measured with cosine similarity between discussions' tf-idf vectors, as is discussed in Section 3.2.1. User's days are only considered if they view more than 5 distinct discussions. As in the user activity grouping, we consider the users with the top $\frac{2}{5}$ highest average daily diversity as having *high initial diversity*. The bottom $\frac{2}{5}$ is considered to have *low initial diversity*.

Recommendation Click Through Rate
Click through rate (CTR) shows us which users can be considered *recommendation followers*. There are some corner cases however, where CTR is not a good indication of whether a user tends to follow recommendations. If a user has been served recommendations twice and has clicked once, they have a CTR of 50% but this is not enough data to classify them as a recommendation follower. To avoid these sorts of cases, we only consider only users who have been served at least 20 recommendations to be in either of the groups. Of the users with more than 20 served recommendations, all those with 0% CTR are classified as *non-followers*. In the remaining users, all users with at least 3 clicks at a CTR of at least 0.5% are considered *followers*. These cutoff points were chosen by considering the spread of number of recommendations served and clicks in our data set.

3.3 Baseline Recommender

Collaborative filtering is one of the most successful techniques for recommender systems [49]. Intuitively, *collaborative filters* analyze relationships between users and

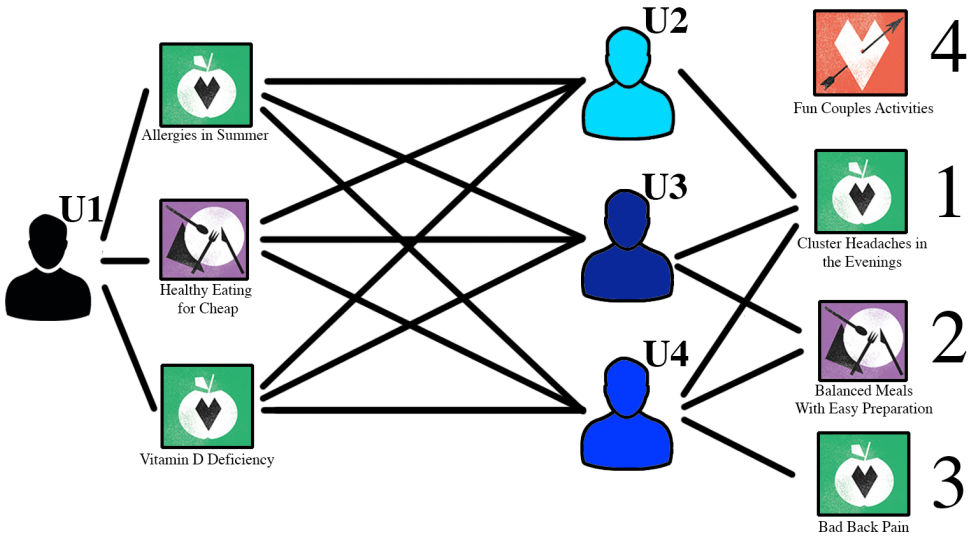


Figure 3.5: Collaborative filtering neighborhood method example: Users with similar preferences (U2, U3, U4) are used to find recommendations for a given user (U1). The final ranking for recommendations for U1 are shown on the right. Black lines indicated preference for an item. Diagram inspired by [30].

interdependencies among products to predict new user-item relationships [30]. There are two main methods of collaborative filtering, neighborhood methods and latent factor methods [30]. *Neighborhood methods* rely on finding similar users or similar items to infer ratings. Figure 3.5 shows a simplified example of the neighborhood method finding similar users to generate recommendations. In the example, black lines link users to discussion items for which they have shown preference for. The three users U2, U3, and U4 have shown preferences for the same items as U1, so their other preferred items are used as candidates for recommendation. The discussions on the right are the recommendations for U1, with numbers indicating the ranking of recommendations.

In our experiment however, we chose to use a *latent factor method* to implement collaborative filtering for our baseline. Latent factor methods are the current state of the art in collaborative filtering. Latent factor-based collaborative filtering works by learning the non-obvious factors (called *latent factors*) to describe patterns in the relationships between users and items. Latent factors characterize both items and users and can be obvious such as "focus on relationships", less obvious like "quirkiness" or totally uninterpretable [30]. All of the latent factors together make up the *latent factor space*, $\mathbb{R}^f$, where f is the number of latent factors. Each user u and item i can then be represented as *latent factor vectors* in $\mathbb{R}^f$ as $y_i \in \mathbb{R}^f$ and $x_u \in \mathbb{R}^f$.

The latent factor vectors are used to generate recommendations by predicting users' ratings for items. A user u's rating r for an item i, is approximated by taking the dot product of the item and user's vectors, as is shown in Equation 3.5 [30].

$$r_{ui} = y_i^T x_u \tag{3.5}$$

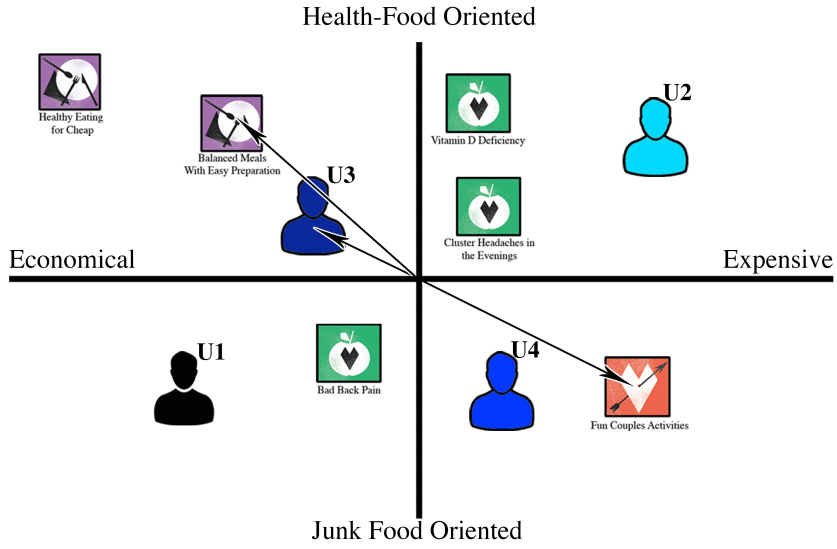


Figure 3.6: Collaborative filtering through latent factors example with 2 latent factors. Algorithm describes users and items as n-dimentional vectors, where n in the number of latent factors. Vector values are learned from patterns in previous user browsing history. In this example, U3 is more likely to enjoy "Balanced Meals With Easy Preparation" than "Fun Couples Activities." Diagram inspired by [30].

Conceptually, the dot product measures how aligned with each other the vectors are. Figure 3.6 shows a toy example of users and discussions defined according to two latent factors. In reality, the number of latent factors would be far greater and would not be named. In our example, we show the latent factor vectors for one user and two dicussions. We would expect user U3 to have a higher rating for the discussion "Balanced meals with easy prep" than the discussion "Fun couples activities", since the former item's vector lines up much more with the user's than the latter item's.

Taking the dot product of latent factor vectors is trivial, the challenge comes when programming an algorithm to learn latent factor vectors for items y_i and users x_u. Many successful implementations of latent factor models use a learning technique called matrix factorization to learn the latent factor vectors [30]. Matrix factorization has become popular due to its scalability, accuracy, and success in the Netflix Prize[2]. In matrix factorization, the algorithm minimizes the regularized squared error given a set of known user-item ratings, also called a training set. In other words, the algorithm tries to set y_i and x_u such that they can predict the known ratings as accurately as possible. The equation for minimizing the error on the known set, or training set, is shown in Equation 3.6. κ is the training set of (u,i) pairs with known ratings and λ is a constant to control the extent of regularization. Regularization is used to avoid overfitting, when a model becomes to sensitive to random noise in the data.

$$\min_{x^*,y^*} = \sum_{(u,i)\in\kappa} (r_{ui} - y_i^T x_u)^2 + \lambda(\|x_u\|^2 + \|y_i\|^2) \tag{3.6}$$

[2]http://netflixprize.com/

We have mentioned ratings in our description of latent factor methods. In recommender systems, ratings are a type of explicit feedback. *Explicit feedback* is explicit input by users regarding their interest or disinterest in products. In many systems, such as ours, explicit feedback is not available and recommenders must work with implicit feedback. *Implicit feedback* is data used to estimate user preference, such as browsing history, purchase history, or any number of other user activities. An important characteristic of implicit feedback is that there is no negative feedback, it is hard to reliably infer which items a user did not like. In our case, the implicit feedback is the number of times the user visited the discussion page. If a user visited a discussion page i three times, $r_{ui} = 3$. Note that with implicit feedback, the value indicates confidence rather than preference. A user might greatly enjoyed reading discussion and only visit once, but search through several pages of another discussion looking for a specific comment. In contrast, a user might surf through several discussions they do not particularly enjoy simply to pass the time. Despite this, the numerical value of implicit feedback is certainly useful, since it tells us about the confidence that we have in a certain rating. Hu et al. [24] have taken the characteristics of implicit feedback and modified Equation 3.6 to work with them.

Here we will describe Hu et al.'s [24] modifications to Equation 3.6. First, the notion of ratings needs to be modified to reflect confidence of user preference rather than measured user preference. To this end, a users preference for a given item is defined as a binary variable p, shown in Equation 3.7 . In terms of our experiment, if a user has viewed a given discussion ($r_{ui} > 0$), we have an indication that they have a preference for that item ($p = 1$). Otherwise, we have no indication of preference.

$$p_{ui} = \begin{cases} 1 & r_{ui} > 0 \\ 0 & r_{ui} = 0 \end{cases} \tag{3.7}$$

Next, we need a way to indicate confidence in a user's preference p for an item. Hu et al. [24] argue that, in general, as r_{ui} grows, we have a stronger indication that the user indeed likes the item. They propose Equation 3.8, where there is a minimal confidence c for every (u,i) pair, but an increase as more implicit evidence for positive preference is observed. α is used as a constant to control the rate of confidence increase.

$$c_{ui} = 1 + \alpha\, r_{ui} \tag{3.8}$$

Another difference introduced by using implicit feedback is that all possible (u,i) pairs can be used as training data, rather than only those with known ratings. Given this difference and the introduction of the notion of confidence, Equation 3.6 is modified as shown in Equation 3.9.

$$\min_{x^*,y^*} = \sum_{(u,i)} c_{ui}(p_{ui} - x_u^T y_i)^2 + \lambda(\|x_u\|^2 + \|y_i\|^2) \tag{3.9}$$

Alternating Least Squares (ALS) is a method of solving Equations 3.9 and 3.6. ALS solves for x and y by fixing one of the two as a constant and recomputing the other to recompute the least-squares problem. The roles are then reversed and recomputed until convergence. With one of the unknowns fixed, the optimization problem becomes quadratic and can be solved optimally [30]. Since the system computes each y_i independently of the other item factors and each x_u independently of the other user

factors, there is an opportunity for massive parallelization of the algorithm [66]. We will discuss our implementation in the following chapter.

3.4 Serendipitous Recommender

For our serendipitous recommender, we use the graph-based "Declustering" algorithm presented by Zhang et al in 2012 [65]. The declustering algorithm is used to boost serendipitous recommendations from a baseline system. A baseline system is considered to be a non-serendipitous recommender. The intuition behind the algorithm is to determine content clusters that a given user is familiar with, and boost items outside of or on the edges of the cluster(s). For example, if a user browses discussion posts about family friendly automobiles, a recommendation outside of the family-friendly automobile cluster of content is more likely to be serendipitous. However, the recommendation must still be relevant to the user to be serendipitous, which is why a standard recommender is used as a starting point. We chose this algorithm because it is not content-based, meaning we do not have to adapt it to Dutch natural language. More importantly, since it is graph based and showed to be successful in producing serendipitous recommendations, we see it to be representative of the state of the art in serendipitous recommender systems.

The basis of the declustering algorithm is a graph of all content items. In the graph, content items are represented as vertices and the similarity between items is represented as weighted edges. User-based Latent Dirichlet Allocation (LDA) similarity is used to calculate the similarity between vertices. We will not go into the inner workings of LDA here, since it is not the focus of our study, but will provide a high-level description.

LDA [8] is typically used in topic modeling for text documents. The algorithm takes a set of documents as input and creates topics as outputs. Topics are distributions over terms in found in the document. Terms that frequently co-occur in documents will have high probabilities in the same topics. Figure 3.7 shows a simplified example of using LDA for topic modeling in text documents. Two topics are calculated based on an input of five scholarly articles. Each term is given a probability in each topic. We can see that Topic 1 is made up of terms related to biology, which frequently co-occur. Similarly, Topic 2 has high probability for words related to computer science. Once topics are defined, each document can be represented as a distribution of topics.

In Zhang et al.'s [65] use of LDA, each content item is a document but the words are users who prefer the item. In turn, this means that topics are distributions over users. Topics can be thought to represent user communities, since they show users who appear frequently together in documents. Further, a document can then be represented as a distribution of user communities. In Figure 3.8, the LDA example has been modified to show the user-community usage of the topic modeling algorithm. We can see that users who appear frequently on the same item end up with high probabilities in the same topics.

Using LDA as described above, we can find how similar documents are based on their preferring user communities. To measure similarity between documents, we calculate the cosine similarity (Equation 3.1) between documents' topic vectors. Intuitively, this is a measure of how similar the user communities are between content

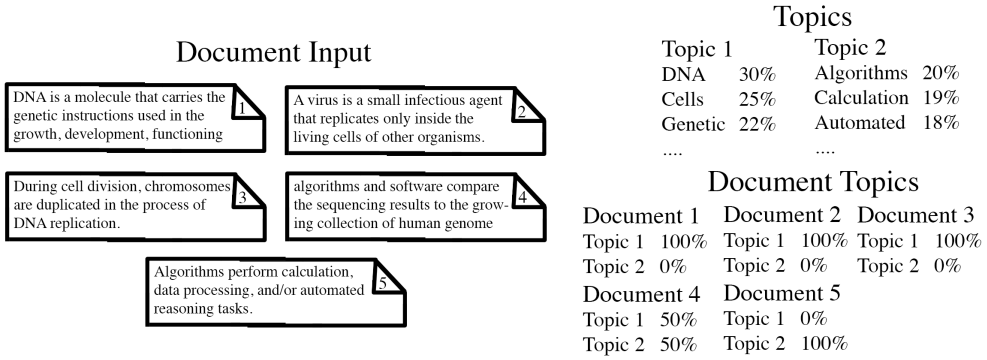


Figure 3.7: Example of traditional usage of LDA

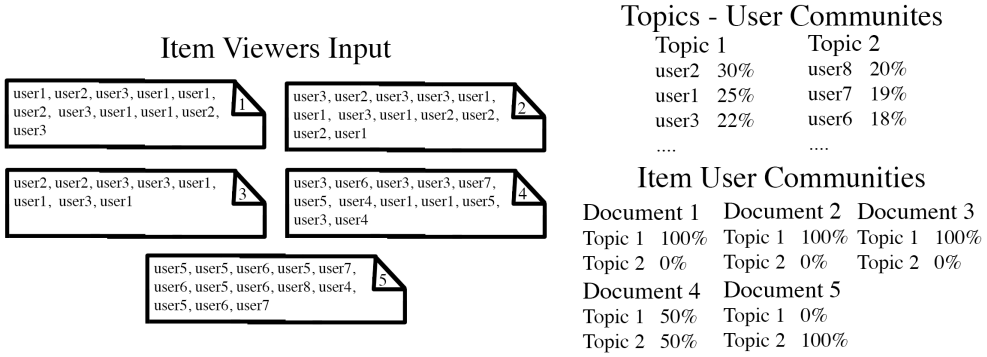


Figure 3.8: Example of user community usage of LDA

items. The similarity is then used to find the weight between vertices in our graph. As explained in Section 3.2.1, cosine similarity's output range between 0 (orthogonal vectors), and 1 (exactly equal vectors). However, less similar items should have greater edge weight, so we converted similarity to distance and used that as the edge weight. Using these edge weights, we can create a graph of all content items where items having similar user communities are clustered together. Formally, this results in a graph $G = (V, E)$, where each vertice $i \in V$ is an artist and edges $(i, j, weight) \in E$ are weighted by $1 - cosineSimilarity(i, j)$.

Once the complete graph is calculated, it can be used to generate serendipitous recommendations. The first step in the algorithm is to generate a *user-subgraph* $G_u \in (V_u, E_u)$ that only includes content items preferred by a given user. Next, a baseline recommender is used to generate a set R of potential recommendations. Each of these potential recommendations R_i is added to the subgraph, and a clustering coefficient is calculated for R_i. The clustering coefficient for each potential recommendation is calculated according to Equation 3.10. Conceptually, the clustering coefficient is the total number of triangles made in the user-subgraph which include the potential recommendation divided by the total number of possible triangles that could exist.

$$Clustering(R_i) = \frac{2|\{e_{jk} : v_j, v_k \in neighbors(i),\ e_{jk} \in E_u\}|}{|neighbors(i)|(|neighbors(i)| - 1)} \tag{3.10}$$

The clustering coefficient shows how clustered the potential recommendation is with the user's subgraph. The least clustered recommendations are boosted to the top, creating a more serendipitous item ordering than the original baseline. We will discuss how we implemented this algorithm for our experiment in the following chapter.

3.5 Hypotheses

With our metrics defined, we can define hypotheses for our research questions from Section 1.1. Our first research question was: *How does serendipity in a recommender systems affect mitigation of the filter bubble effect?* We have broken this research question into four hypotheses, listed below. The first two hypotheses regard the diversity of content recommended. We expect the baseline recommender to learn more about the user as the experiment goes on and therefore recommend a narrowing set of items. In contrast, we expect the serendipitous recommender to recommend increasingly diverse sets of items to users. The second two regard the other part of the filter bubble, the change in content consumed. We hypothesize that the serendipitous recommender will help users discover new content and thus increase the diversity of content consumed over the testing period. In contrast, we expect the baseline recommender to cause users to have constant diversity of content consumed.

RQ1-H1: Users receiving *baseline* recommendations will have narrowing diversity of content *recommended* to them over the testing period.
RQ1-H2: Users receiving *serendipitous* recommendations will have an increase in diversity of content *recommended* to them over the testing period.
RQ1-H3: Users receiving *baseline* recommendations will have negligible change in diversity of content *consumed* over the testing period.
RQ1-H4: Users receiving *serendipitous* recommendations will have an increase in diversity of content *consumed* over the testing period.

Our second research question was: *What user characteristics affect mitigation of the filter bubble effect?* We have broken this research question into three hypotheses, one for each user characteristic we investigate. First, we expect users with higher initial content diversity to experience less of a filter bubble effect in the content they consume than users with lower initial content diversity. Our reasoning is that users who already consume a diverse set of content will more readily expand their horizons. Second, we expect users with higher click through rate on the recommenders to experience less of a filter bubble effect in the content they consume. This is inspired by Nguyen et al.'s findings [38] that users who used the recommender system experienced less of a narrowing of diversity of content consumed. Lastly, we expect users who are more active to experience less of a filter bubble effect in the content they consume. Similar to the first hypothesis, our reasoning is that users who spend more time on the site will more readily expand their horizons.

RQ2-H1: Users' initial diversity of content consumed directly correlates with expansion of diversity of content consumed.
RQ2-H2: Users' click rate on recommendations directly correlates with expansion of diversity of content consumed.

RQ2-H3: Users' activity level directly correlates with expansion of diversity of content consumed.

In this Chapter, we have described the conceptual approach we are taking in our research. Next, we discuss the implementation details in making our research a reality.

Chapter 4

Implementation

One of the contributions of this research is the fact that it is performed on a live platform with real users who are unaware of the experiment. This aspect also adds some complexity to the experiment. We must guarantee recommendations to be served quickly enough to load with the rest of the page, which typically responds to requests in under 100ms. Further, the system must have constant up-time, and handle possibly large volumes of requests. In this section, we will describe our experiment setup and implementation of the algorithms detailed in the previous chapter.

4.1 Data Pipeline and Retraining

The first challenge we faced in implementing our project at Sanoma was selecting the correct users to use in our experiment. Since users are tracked using cookies, there is a lot of noisy data from one-time visitors, users who block cookies, or users who frequently clear cookies. Our experiment required us to find persistent cookie ids, *traceable users*, who we could train models for and then expect to reappear during our testing period. To find traceable users, we sampled 1 month of data and selected users who appeared on more than five separate days and had viewed at least ten different discussions throughout the month. In preliminary investigations, we found that 83% of traceable users found in this way would return in the month following the sampling month. This is an acceptable percentage.

For the initial setup of our experiment, we found 16,909 cookie ids as traceable users. The traceable users' view event data from the month preceding the experiment was used as the intial training data for our recommenders. Then, each day as the experiment ran, we added a new day of view event data for the models to retrain on.

We created a pipeline to move the view-event data from Sanoma into our platform. This pipeline was used for both the initial data import and the daily updates. We broke the daily process into four steps, shown in Figure 4.1. In this section, we will describe each of the four steps.

(1) At Sanoma, the user event data is stored on a Hadoop[1] big data cluster. However, not all of the event data is needed for our recommendation engine. We are only interested in view event data from our traceable users' activity on discussion pages on the Viva Forum website. Thus, we filter out events from other platforms, events from

[1] http://hadoop.apache.org/

Figure 4.1: Data pipeline diagram

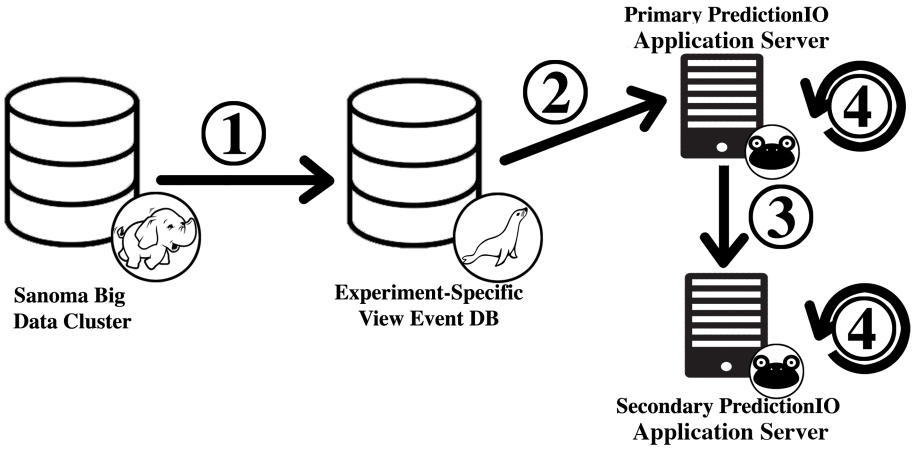


Table 4.1: Example raw view event data from Sanoma big data cluster

cookie id	ts	content_id	ip/useragent	session	viewport
qcbdg0acg	1469604760	/forum/geld-recht/scheiden/list_messages/336663	***	cx4xznjprx	360x616
qneikcod0	1469604747	/forum/relaties/vreemdgaan/list_messages/336729	***	rf6n6pdjho	1301x641
qcbdg0acg	1469604679	/forum/geld-recht/scheiden/list_messages/336663	***	cx4xznjprx	360x616
qmmjydak4	1469603085	/forum/relaties/vreemdgaan/list_messages/336729	***	54dmkel4rz	956x889
qm2l6843w	1469603069	/forum/kinderen/klassenouder/list_messages/333023	***	eoc4cqwbnu	360x512

non-content pages on the Viva Forum, and events from other users. An example set of raw view event data from traceable users is shown in Table 4.1. Once we have a set of traceable users, we can pull their Viva Forum discussion page view data from the Sanoma cluster and on to our own MariaDB[2] server. MariaDB is an open-source fork of the MySQL relational database management system. We used MariaDB rather than MySQL because it was highly recommended by Sanoma staff.

(**2**) Once the data is on our MariaDB server, we process the raw event data into the format required by our recommendation engine. Most importantly, we convert messy content_id strings into content identification numbers, indicating individual discussion posts. Additionally, when new discussions are found in the events, we scrape the Viva Forum site to get the title, category, and creation date. Finally, the data is exported in JSON format. Each view event is converted to JSON, as is shown in Listing 4.1. Further, each distinct discussion item found in events is converted to JSON, as is shown in Listing 4.2. Once the view events data has been processed and converted to JSON on the MariaDB server, we import them into our recommendation engine server to retrain the models.

[2]https://mariadb.org/

Listing 4.1: JSON describing first view event from Table 4.1

```
{
  "event":  "view",
  "entityType": "user",
  "entityId": "qcbdg0acg",
  "targetEntityType": "item",
  "targetEntityId": 336663,
  "eventTime": "2016-07-27 07:32:40"
}
```

Listing 4.2: JSON defining item from first view event from Table 4.1

```
{
  "event":"$set",
  "entityType":"item",
  "entityId":"336663",
  "properties":{
    "category":"Geld en Recht",
    "title":"Scheiden",
    "date_created":"2016-07-26"
  }
}
```

(**3**) We use PredictionIO[3] as our recommendation application framework. PredictionIO is an open-source machine learning server, allowing for the deployment and retraining of machine learning algorithms as web services. Using PredictionIO's framework, we are able to train a model with new data while continuing to serve recommendations. We balanced the recommendation load across two PredictionIO applications servers, only one of which would be used to retrain models. The PredictionIO server used to retrain the models is referred to as the *primary recommendation application server*. As soon as a new model is computed on the primary server, it is sent to the secondary server.

(**4**) Finally, both recommendation engines redeploy the latest model available to them. We separate the retraining from redeployment steps to keep both recommendation engines on the same model. We repeat steps 1-3 every night during the experiment. Step 4 is repeated every hour, meaning that the newly trained model in step 3 is deployed on both recommender application servers on the first hour after it is ready. Step 1 was scheduled using Jenkins[4] and Steps 2, 3, and 4 were scheduled using the Unix Crontab[5].

In addition to scheduling the jobs, we had to monitor our recommendation engine servers. We used a service monitoring package called Monit[6]. One of our contributions to the PredictionIO open source project was writing a guide on how to use Monit with PredictionIO[7]. In addition to monitoring the memory and storage usage of our engines, we were also use Monit to quickly recover from any crashes or errors that took down the recommender service.

4.2 Experiments

4.2.1 Pilot Experiments

We had two pilot experiments to try out the Viva Forum as a domain and test our pipelines before starting a large-scale experiment. The pilot experiments ran while we were still developing our serendipitous recommender, so we only used the baseline system. Both pilot experiments had only a few hundred users.

The primary purpose of the first pilot was to test the data pipeline and request pipeline in a live setting. We used a test group of about 100 users and put the recom-

[3] https://prediction.io/
[4] https://jenkins.io/
[5] https://en.wikipedia.org/wiki/Cron
[6] https://mmonit.com/monit/
[7] http://predictionio.incubator.apache.org/deploy/monitoring/

mendation panel at the bottom of Viva Forum discussion pages. We had some issues with our recommender servers, prompting us to make some additions to the scripts to make them more robust. Besides the server uptime, we were not satisfied with the amount of clicks our recommender was getting. We reasoned that the bottom of the discussion item page could be a poor location for two reasons. First, users would have to scroll all the way to the bottom to see it. Second, the typical user flow was to find discussions at the home page and navigate from there, rather than from discussion to discussion.

With the low performance of the discussion page recommender in the first pilot, we placed the recommendation panel on the homepage for the second pilot. We used a small test group of 200 users, but this placement gave a satisfactory amount of click traffic and was used for the primary experiment. In Figure 4.2, our recommendation panel is shown on the home page. Our recommendations are displayed in the four-panel row underneath the "Ook Interessant" header, which means "Also Interesting" in Dutch. We ensured that the panel was fully responsive including scaling to mobile device screen size.

4.2.2 Primary Experiment

For our primary experiment, we used the same recommendation panel as is shown in Figure 4.2. This time, we used the maximum amount of users possible given our selection criteria. To select users, we found 16,909 traceable users, as discussed in Section 4.1. Of the 16,909, 9,524 were seen in our experiment. We served recommendations to approximately 5,000 users every day. Figure 4.3 shows how the number of distinct users changed over the course of the experiment. Due to the fleeting nature of browser cookies, we expected and saw a slow drop in participating users over the course of the month. Had the experiment continued for longer than a month, this would have become an issue. We served approximately 45,000 recommendation panels every day, with a similar slow drop-off through the course of the experiment. From a user participation perspective, we consider our primary experiment to be successful.

4.3 Recommender Engine

As mentioned in the previous section, we used PredictionIO to implement our recommender application. PredictionIO makes deploying machine learning algorithms as services as straightforward as possible. There are three levels to a PredictionIO application, the application, engine, and algorithm. The *application* is the software application that is using PredictionIO, allowing for requests and taking in new data needed for the recommender. The *engine* is a logical identity that an application can interact with via the API, acting as a wrapper around the algorithms below it. The *algorithms* are actual computation code that generate prediction models. In our case, we have a single application and engine that contains the algorithms for both serendipitous and baseline recommendations.

Figure 4.4 shows how a recommendation request occurs in our system. When a user visits the Viva Forum homepage, the page is loaded from the normal Sanoma web server (**1**). The page returned by Sanoma also includes a PHP script stored on our PredictionIO application servers. The PHP script reads the user's Sanoma cookie ID,

Figure 4.2: Homepage experiment panel. Our recommendation panel is under the "Ook Interessant" header.

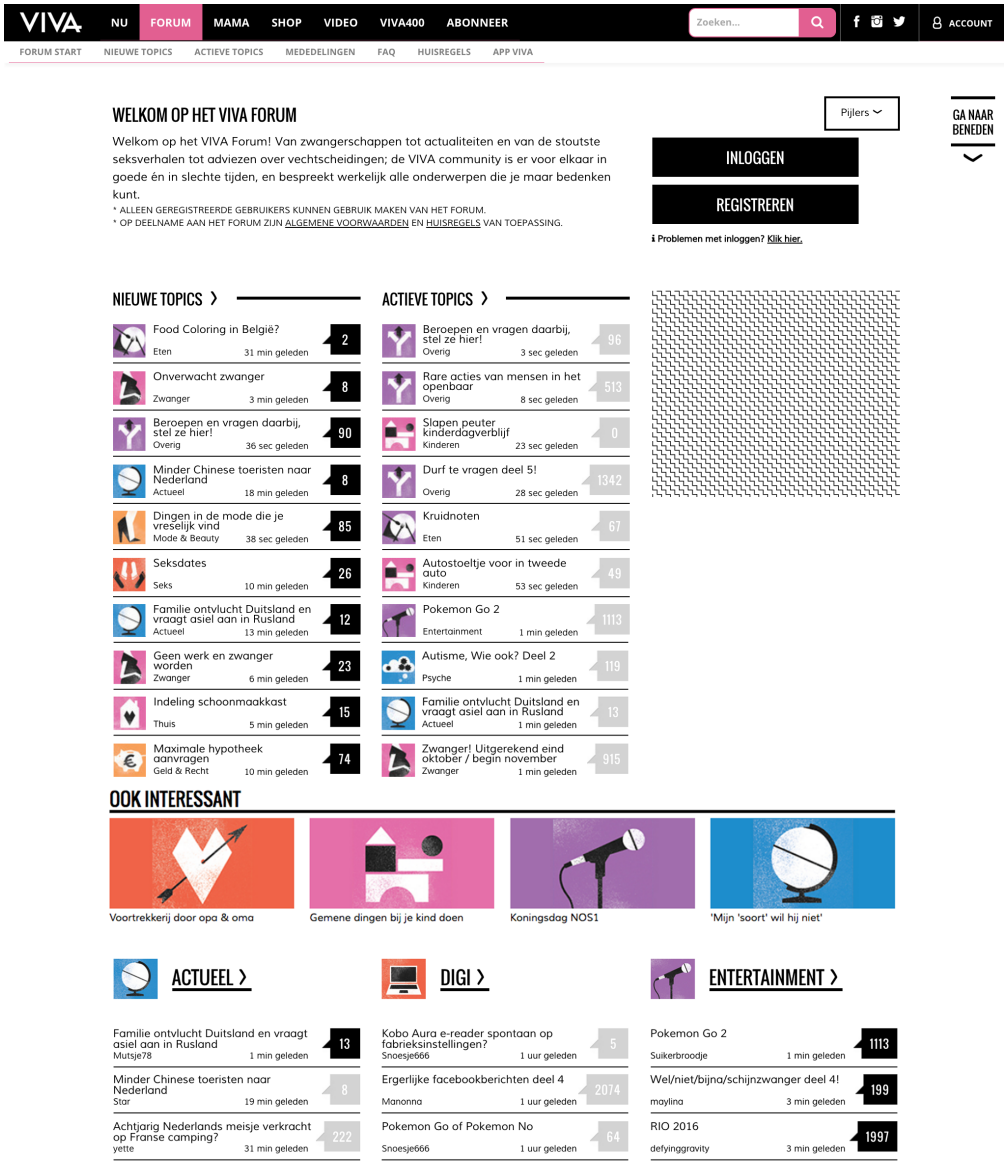


Figure 4.3: Distinct users seen over the course of our primary experiment.

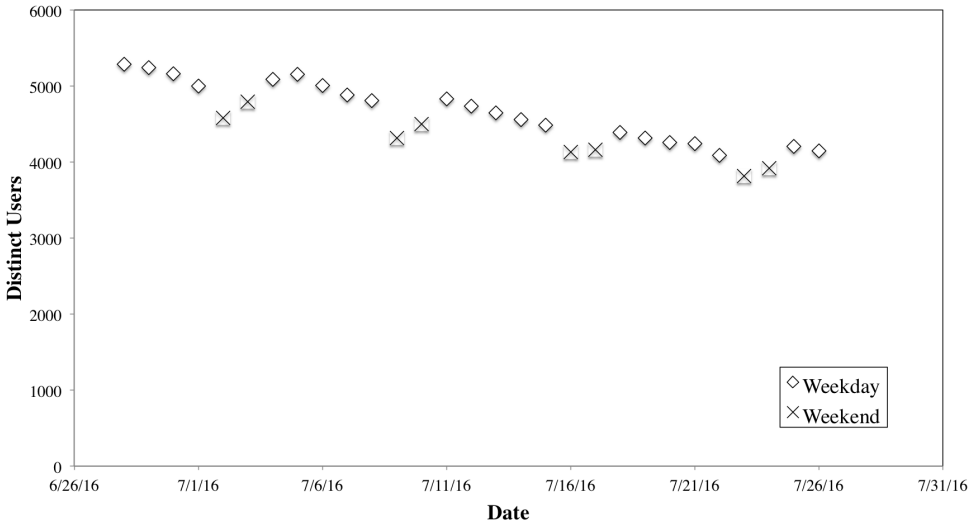
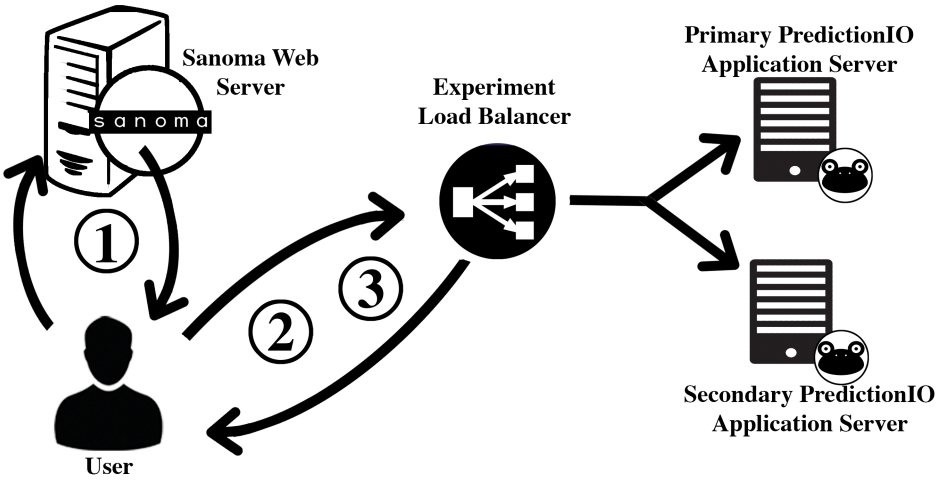


Figure 4.4: Query request diagram



checks it against a list of participants from both test groups, and outputs a Javascript block if the user is part of our experiment (**2**). The Javascript block makes an AJAX call to the PredictionIO servers (**3**) to get recommendations and adds the recommendation panel to the page. The style of the panel will be discussed in the Experiment section. The format for the query from step 3 is shown below, containing the userid, number of recommendations, and the recommender type. The query response is a set of recommended discussions.

Listing 4.3: Query structure for step 3 in Figure 4.4

```
{"user": <cookie id>, "num": 4, "recommender": <baseline or serendipitous>}
```

Like news articles, discussion forum posts often lose their appeal as they age. The original poster may their question answered, or the topic is out of date. We found that, in a month dataset of view events, 43% of views were on items a day or less old and 75% on items less than a month old. We decided to limit both algorithms to recommend dicsussions at most two weeks after their creation date. This way we would not be giving users old recommendations they are not interested in, yet give our recommenders over 1000 items to choose from each day.

The algorithms in PredictionIO are built on Apache Spark[8]. Apache Spark is an open source cluster computing framework, which has many useful libraries built on top of it. One of these libraries is the Machine Learning Library[9] (MLlib). MLlib contains many algorithms, including the ALS collaborative filtering for implicit feedback algorithm discussed in our approach section [24]. Another Spark library is GraphX[10], used for graph parallel computation. We made use of both of these libraries in implementing our recommenders algorithms.

4.3.1 Baseline Recommender

Here we will briefly cover our implementation of the baseline recommender. We used the Apache Spark implementation of collaborative filtering[11] for our baseline recommender. This implementation uses the ALS matrix factorization method for implicit feedback detailed in the Section 3.3. Every time the recommendation engine was retrained in Step 3 of our pipeline in Figure 4.1, the latent factor vectors were recalculated to include the newest day's worth of data. The recommendations could then be quickly calculated upon request.

When using the ALS matrix factorization algorithm, there are a number of parameters to consider and optimize. These parameters are shown in Table 4.2. *Rank* is the number of latent factors to be used. *Iterations* is the number of iterations of ALS to run. *Lambda* (λ) and *Alpha* (α) were discussed in Section 3.3. We used the values suggested by Spark documentation[12] for iterations and lambda, but used Precision@K evaluation to find optimal values for rank and alpha.

Precision in information retrieval is the fraction of retrieved documents that are relevant [34]. *Precision@k* is an evaluation metric for information retrieval which measures the precision of recommendations at a given recommendation set size [34]. When using a precision metric for evaluation, it is necessary to know which documents are considered relevant. In our case, we considered documents that had been viewed by the user as relevant. Using this combination of recommended parameters and parameter estimation through Precision@k evaluation, we are confident in our baseline implementation.

[8] http://spark.apache.org/
[9] http://spark.apache.org/mllib/
[10] http://spark.apache.org/graphx/
[11] spark.apache.org/docs/latest/mllib-collaborative-filtering.html
[12] http://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.mllib.recommendation.ALS

Table 4.2: The parameters for ALS matrix factorization and the methods used to assign their values

Parameter	Description	Method of Assignment
Rank	number of latent factors to use	Used best value according to Precision@k
Iterations	number of iterations of ALS	Used recommended value above 20
Lambda	regularization factor	Used recommended value of .01
Alpha	confidence parameter	Used best value according to Precision@k

4.3.2 Serendipitous Recommender

As discussed in Section 3.4, we used the declustering serendipitous recommender proposed by Zhang et al. [65]. The algorithm uses a content graph where item similarity is measured by a user-based LDA method. Each user has a user-specific subgraph, containing items from that user's history and the edges connecting them. A baseline system is used to generate N potential recommendations, each of which is added to the user-specific subgraph. The recommendations that cluster the least with the user-specific subgraph according to Equation 3.10 are boosted to the top. As far as we know, this algorithm has not been deployed in a live setting before.

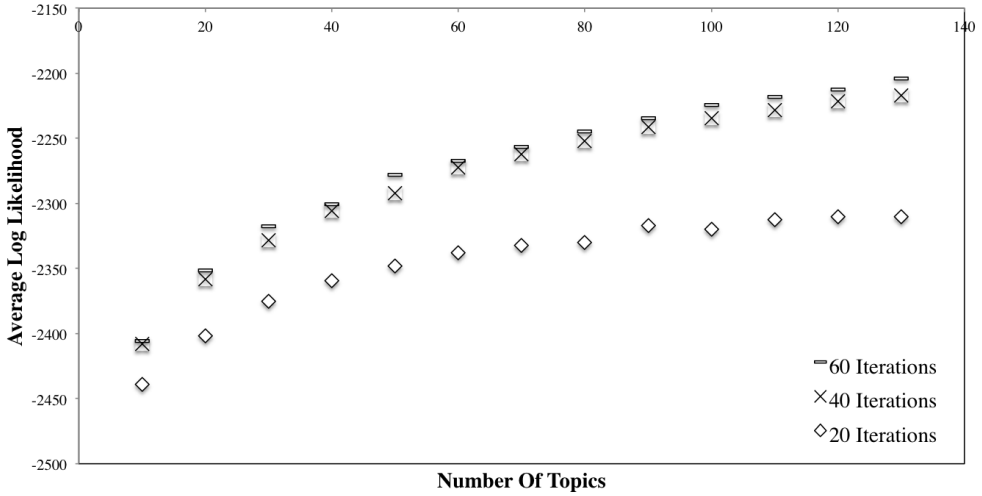
The first challenge in implementing our serendipitous recommender algorithm is creating the full content graph. There is some pre-processing required to set up the discussion items and preferring users as documents and words, so they could be used as inputs for the LDA algorithm. To keep very unpopular items out of the graph, we removed content items with less than three distinct viewers. Also, to keep some indication of strong user preference, if a user viewed a content item multiple times, their user id would appear multiple times in the corresponding document.

With the documents ready, using LDA to get topics and topic distributions was the next step. We made use of Apache Spark's LDA implementation[13], but still had to tune the LDA parameters. We evaluated the LDA algorithm on the month of training data data to see which created the best model. The two parameters we were concerned with were number of iterations and number of topics. As discussed in [8], the optimal parameters for LDA for a given set of data can be found by finding those that maximize log likelihood. Figure 4.5 shows the results of our evaluation. We can see that the benefit of higher iteration values decreases once it is higher than 60. Further, the log likelihood increases greatly between 1 and around 60 topics, but afterwards the gain of adding more topics is not as noticeable. Given these observations, we ran our model with 100 topics and 60 iterations.

With LDA, we are able to generate topic vectors for each discussion (vertice) and use cosine distance ($1 - cosineSimilarity$) to find the edge weights between vertices. This leaves us with a nearly fully connected graph, since most term vectors have non-zero cosine distance. In Zhang et al. [65], edges are trimmed for each user's subgraph according to user-specific average edge weight. Since we had to retrain this algorithm in a live setting for multiple thousand users with limited resources, user-specific edge trimming was too time-consuming. Instead, we used a constant graph density to set a cutoff point for edge weights. We found that a graph density of 15% produced enough

[13] http://spark.apache.org/docs/latest/mllib-clustering.html#latent-dirichlet-allocation-lda

Figure 4.5: LDA Least Likelihood, used to find the ideal number of topics for LDA. Beyond 100 topics and 60 iterations, the added gain to least likelihood of increasing either parameter decreases.



interconnectedness to find clusters in user history. Graph density D is calculated with the Equation 4.1, where $|E|$ is the number of edges and $|V|$ is the number of vertices. After calculating all edge weights, we sorted them and kept the $|E|$ largest weights required to meet the density constant. This gives us our content graph, which we store as a GraphX graph object.

$$D = \frac{2|E|}{|V|(|V|-1)} \tag{4.1}$$

With the complete content graph, we are able to boost baseline recommendations to favor serendipitous items. We use our own experiment's baseline as the baseline model for our serendipitous recommender system. This differs from Zhang et al. [65], who use their own LDA-similarity based baseline. We tried to implement Zhang et al.'s algorithm [65] so it could create each subgraph and check clustering coefficients upon receiving recommendation requests, but were not able to. The primary issue we ran into during implementation was GraphX, which is optimized to distribute a single large graph across memory, rather than many small graphs. This meant that we were not able to calculate all subgraphs simultaneously, leading to far too slow calculation. Our workaround was to use GraphX only for saving and loading graphs and writing our own cluster coefficient function. Even with our workaround, the calculations were too slow to be done on-the-fly with requests. Instead, we precalculated all serendipitous recommendations with each daily retraining of the model. This way, when the application requested a serendipitous recommendation, only a lookup was needed.

4.3.3 Evaluation

We evaluated our recommenders before launching the experiment, but continued the evaluation with each daily retraining of the models show in the pipeline in Figure 4.1.

The purpose was to evaluate the recommender models throughout the testing period. On each day of the experiment, we generated the baseline and serendipitous recommendations for each of our users regardless of test group. Note that this is separate from the experiment data because we are ignoring how users interact with the system, only observing how the models act through the experiment.

Before evaluating the serendipity of recommendations, we evaluated whether the baseline and serendipitous recommender were producing different recommendations. Calculating the difference between the two recommendations is important because it allows us to observe how much effect our boosting algorithm had on the final recommendation, if any. We calculated the ordered and unordered overlap between of the top 4 items of each recommendation system, since our recommendation panel has 4 items. Unordered overlap (Equation 4.2) is the percent intersect between the two recommendation sets. Ordered overlap (Equation 4.3) only considers items to be overlapping if the occupy the same rank in the recommendation sets. Suppose a user has a baseline recommendation of **[item1, item2, item3, item4]** and a serendipitous recommendation of **[item1, item3, item5, item6]**. These two recommendations would have an ordered overlap of 25% and an unordered overlap of 50%.

$$unorderedOverlap = \frac{|SerendipRec \cap BaselineRec|}{4} \tag{4.2}$$

$$orderedOverlap = \frac{\sum_{i=1}^{4} |SerendipRec_i \cap BaselineRec_i|}{4} \tag{4.3}$$

In the Figure 4.6, we show the average ordered and unordered overlap between the baseline and serendipitous recommendations generated on each day we retrained the model. Overlap closer to 0% indicates that the serendipitous and baseline recommenders are serving completely different recommendations. The unordered overlap stayed around 30% for most of the experiment and the ordered overlap was between 15% and 20%. We find this to be acceptable overlap, especially considering the constraint of only recommending items created in the last two weeks.

To ensure that our serendipitous recommender was performing as expected, we evaluated the serendipity of the recommendations generated by the model against the baseline on each day of the experiment. We expected to see the serendipitous recommender to have consistently higher serendipity than the baseline. As discussed in the Section 3.2.2, we used distance from user profile as our metric for serendipity. Figure 4.7 shows the results of our evaluation for serendipity over the testing period. As we expect, the serendipitous recommender had consistently more serendipitous recommendations than the baseline. We also observe a slight increase in distance from the user profile over time from both recommenders. We did not investigate the increase extensively however, since the serendipitous recommender remained consistently more serendipitous.

We also performed spot checking to compare baseline and serendipitous recommendations. As an example, consider a user from the baseline group "Bob" who has viewed 441 distinct forum threads. 350 of the distinct items come from the categories *Relationships*, *Children*, and *Fashion-Beauty*. Bob's baseline recommendation on halfway through our experiment consisted of three *Relationship* discussions and a single *Children* discussion. In contrast, his serendipitous recommendation would have

Figure 4.6: Recommendation algorithm overlap between the baseline and serendipitous algorithms over the testing period

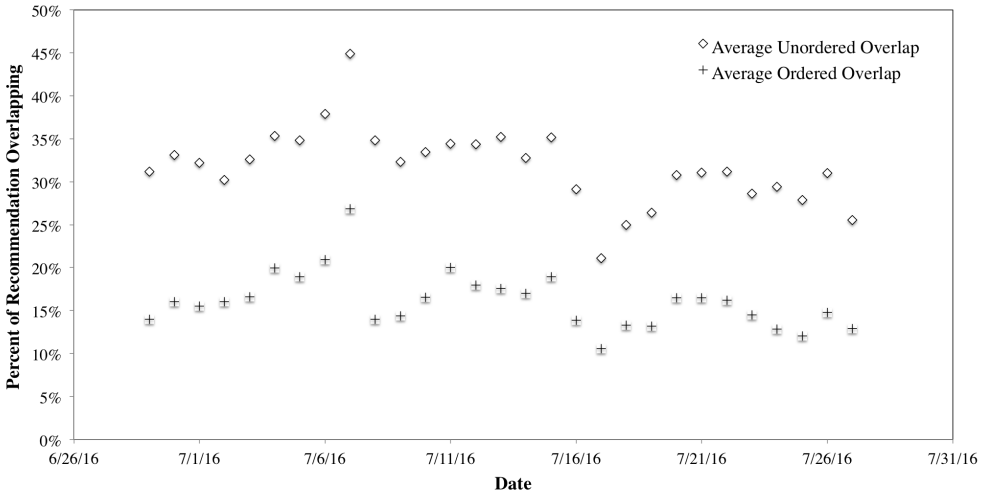
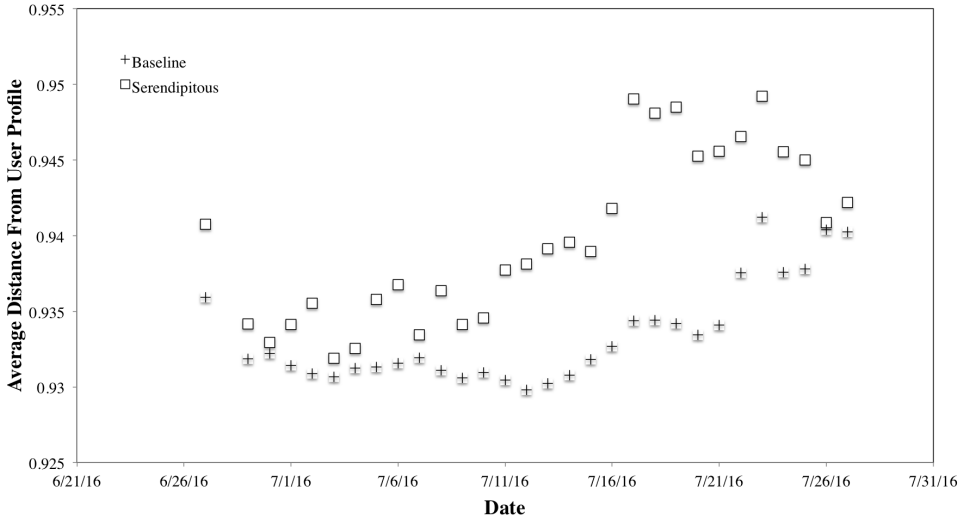


Figure 4.7: Recommendation distance from user profile over testing period



included items from four different categories, with only one item coming from his historical top three most favored. With the recommenders performing as we need and the recommendation applications working, we can move on to analyzing the data from our experiment.

Chapter 5

Results and Discussion

We have broken up our results into two sections, one for each research question. For each research question, we describe how we calculated the results required to test our hypotheses. The results are presented in graphs, allowing for easier interpretation. The graphs are used to test our hypotheses, the results of which are summarized in a table at the end of each section. Following the results, we present some discussion of the results in Section 5.3.

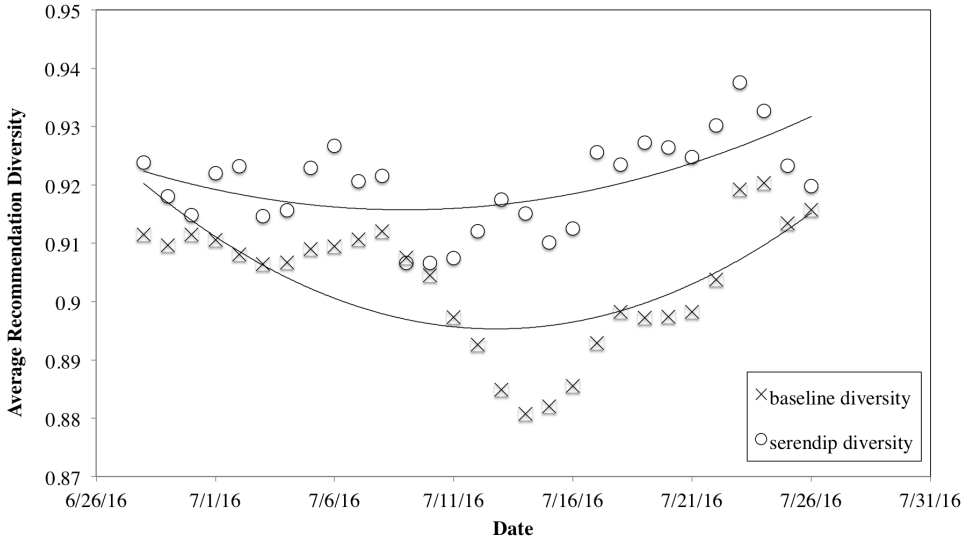
5.1 Serendipity and Filter Bubble

Our first research question was: *How does serendipity in a recommender systems affect mitigation of the filter bubble effect?* To answer this, we deployed a standard collaborative recommender system and a serendipitous recommender system on our live discussion forum website. We served recommendations to 9,524 distinct users over the course of our month-long experiment, with 4,776 always receiving standard recommendations and 4,748 always receiving serendipitous recommendations. As described in Section 3.2.1, we measure the filter bubble effect by looking at the diversity of content consumed and diversity of content recommended for each user on each day of our experiment.

First, we will investigate the **diversity of content recommended** to each recommender type user group. To calculate diversity of content recommended, we start with each user's distinct recommendation sets served for each day. A *recommendation set* is a 4-item recommendation. Since the model retrains and redeploys once per day, each user has at most 2 distinct recommendation sets served to them each day. Combining all users distinct recommendation set left us with between 2,000 and 3,000 recommendation sets per recommender type per day. Then, the diversity of each recommendation set is taken using Equation 3.2. Finally, the diversity values are grouped by recommender type and day and averaged, giving us each recommender's daily diversity. The average daily diversities of content recommended are shown in Figure 5.1. Polynomial trendlines are included for both data series.

With the results illustrated in Figure 5.1, we can test our 1st and 2nd hypotheses from our first research question defined in Section 3.5. Over the testing period, we expected users receiving baseline recommendations to have narrowing diversity of content recommended, while expecting users receiving serendipitous recommendations

Figure 5.1: Average daily diversity of content recommended to individual users, separated by recommender type.



to have increasing diversity. There is a trend in increased diversity in the serendipitous recommendations, but the baseline also has an upward trend. It is noteworthy that the baseline recommendation diversity has a far more pronounced dip in the first half of the experiment period than the serendipitous recommendations. We accept our hypothesis that the serendipitous user group would receive increasingly diverse recommendations but still reject our hypothesis that the baseline user group would have decreasing recommendation diversity.

Next, we investigate the **diversity of content consumed** by users separated by their recommendation type. In this case, we consider the content consumed by individual users on each day of the experiment. Again, we use Equation 3.2 to measure the diversity of each user's set of daily consumed content. To avoid random noise in the data we only consider the *(user, day)* pairs when the user views at least six items on the given day. This left us with between 1,000 and 1,500 users' daily browsing history for each recommender on each day. We group the diversity values by recommender type and day, giving us each day's average diversity of content consumed separated by recommender group. The daily diversities of are shown in Figure 5.2.

With the results illustrated in Figure 5.2, we can evaluate our 3rd and 4th hypotheses from our first research question. Over the testing period, we expected users receiving baseline recommendations to have negligible change in diversity of content consumed, while expecting users receiving serendipitous recommendations to have increasing diversity of content consumed. There much less of an obvious trend in diversity of content consumed in both recommender type groups. In fact, the baseline and serendipitous recommenders each have more diverse content consumption than the other at some point in the experiment. We will discuss this finding further in Section 5.3. Further, the range of diversity values here is far smaller than in the content recommended graph. That being said, we accept our hypothesis that users receiving baseline recom-

Figure 5.2: Average daily diversity of content consumed by individual users, separated by recommender type.

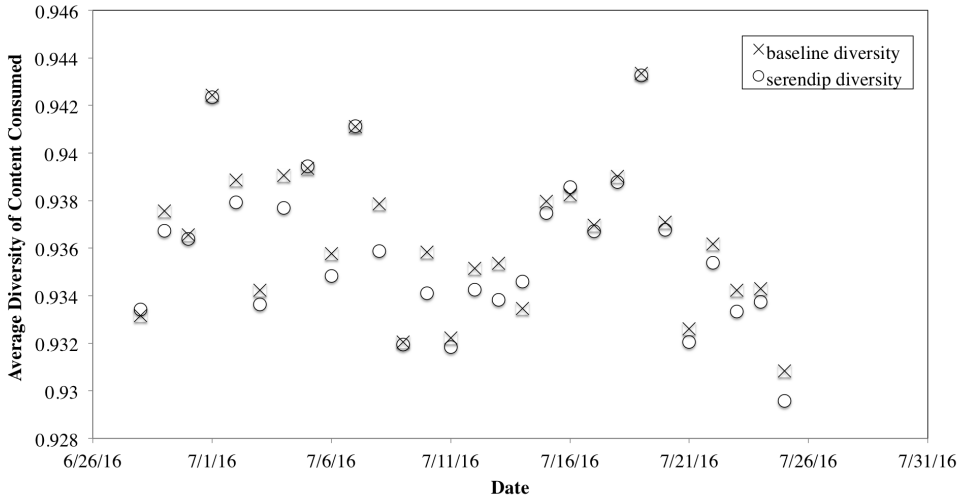


Table 5.1: Hypothesis tests for first research question

Hypothesis	Accept/Reject
Users receiving baseline recommendations will have narrowing diversity of content recommended to them over the testing period.	reject
Users receiving serendipitous recommendations will have an increase in diversity of content recommended to them over the testing period.	accept
Users receiving baseline recommendations will have negligible change in diversity of content consumed over the testing period.	accept
Users receiving serendipitous recommendations will have an increase in diversity of content consumed over the testing period.	reject

mendations would have negligible change in diversity of content consumed. However, we reject our hypothesis that users receiving serendipitous recommendations would have an increase in diversity of content consumed.

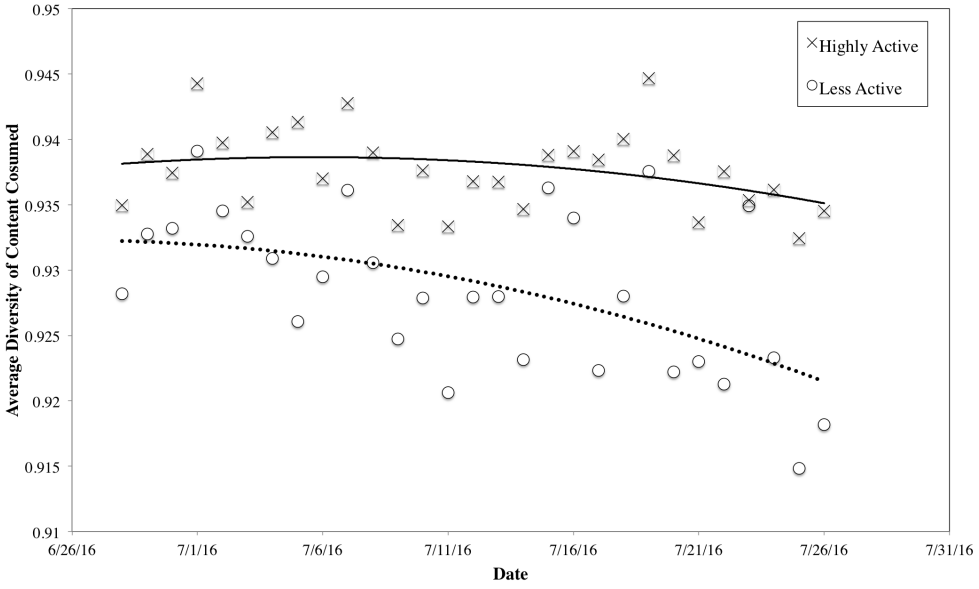
The results of our hypothesis testing are summarized in Table 5.1. We will discuss these results in the last section of this chapter, after testing the hypotheses from our second research question.

5.2 User Characteristics and Filter Bubble

Our second research question was: *What user characteristics affect mitigation of the filter bubble effect?* The user characteristics we investigate are user activity, initial user content diversity, and recommendation click through rate. To answer this, we divide the users into groups based on the metrics for each characteristic defined in Section 3.2.3. We then compare the user groups' daily consumed content diversity. Since we are no longer interested in the role of serendipity, we only use users from the baseline recommendation group in these tests.

First, we consider the user's **activity level**. After finding users' activity levels according to their average daily views, we separated out a group of 1,895 highly active

Figure 5.3: Showing average individual users' daily diversity of content consumed over the experiment, separated by recommender type and activity level grouping. We only consider users from the baseline recommender group and only data from *(user,day)* pairs where the user viewed at least 6 discussions.



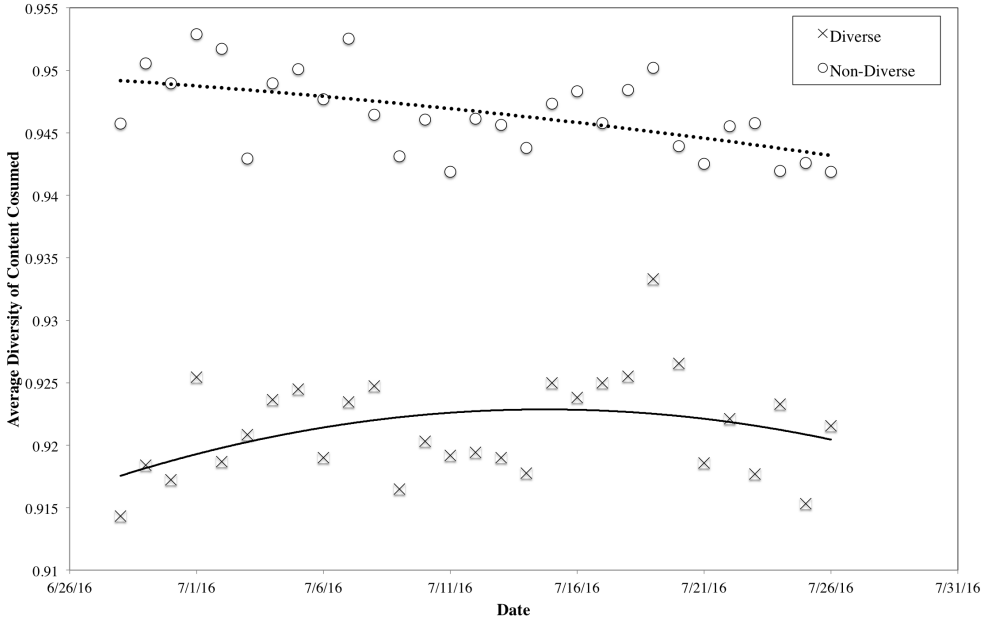
users and 1,896 less active users. Note that we left out the middle 960 "averagely active" users, as described in Section 3.2.3. Again, we consider only *(user, day)* diversity values where the user views at least six items. The remaining values are grouped by day, activity level classification, and recommendation type then averaged. Figure 5.3 shows our results in comparing user activity to diversity of content consumed. Polynomial trendlines are included for both data series.

We hypothesized users' activity level would directly correlate with expansion of diversity of content consumed. In other words, the more active users would consume a more diverse set of content which would expand over the duration of the experiment. In the data represented in Figure 5.3, we observe that highly active users maintained a higher diversity level throughout the experiment. Also, both user groups experienced slight drops in diversity over time. However, the active users' drop in diversity is far less pronounced as it is nearly flat. Therefore, we accept our hypothesis that user activity level correlates with higher diversity of content consumed, and that the narrowing of diversity is less pronounced in active users.

Next, we study users' **initial consumption diversity**. In this case, we separated users into groups according to the diversity of content they consumed in the month before our experiment began. Our resulting user groups are 1,717 users with diverse pre-experiment consumption, 1,746 with nondiverse pre-experiment consumption, and 830 with average diversity. Figure 5.4 shows the diversity of content consumed of each user group during the experiment. Polynomial trendlines are included for both user groups.

We hypothesized that users' initial diversity of content consumed would directly

Figure 5.4: Showing average individual users' daily diversity of content consumed over the experiment, separated by recommender type and initial diversity grouping. We only consider users from the baseline recommender group and only data from *(user,day)* pairs where the user viewed at least 6 discussions.



correlate with expansion of diversity of content consumed. From the data in Figure 5.4, we observe that users with higher initial diversity of content consumed maintain their higher content diversity through the experiment period. The diversity remains nearly constant for both groups though, particularly with the non-diverse user group where it fluctuates up and downwards. It is shown that users with higher initial diversity stay there, but if anything their diversity of content consumed drops more than the second user group. This being the case, we reject our second hypothesis that users with higher initial diversity would have more of an expansion of diversity of content consumed.

Finally, we examine if users with higher **click through rate** experience less of a filter bubble effect than users who ignore recommendations. In this case, we had 1,105 users we considered to be "recommendation followers" and 3,800 we deemed to be ignoring the recommendations. Figure 5.5 shows our results for user CTR versus diversity of content consumed. The user group with low CTR is dubbed as *Non-Followers* and users with higher CTR are *Followers*.

We hypothesized that users' click rate on recommendations directly correlates with expansion of diversity of content consumed. In other words, followers would experience less of a filter bubble effect than those who did. Our analysis shows no connection between CTR and consumed content diversity. No group consistently shows higher diversity of content consumed. Further, the differences in diversity values are far lower than when grouping users by activity level or initial content diversity. Thus, we reject this hypothesis as well.

We have summarized the results of hypothesis testing for our second research ques-

Figure 5.5: Showing average individual users' daily diversity of content consumed over the experiment, separated by recommender type and follower grouping. We only consider users from the baseline recommender group and only data from *(user,day)* pairs where the user viewed at least 6 discussions.

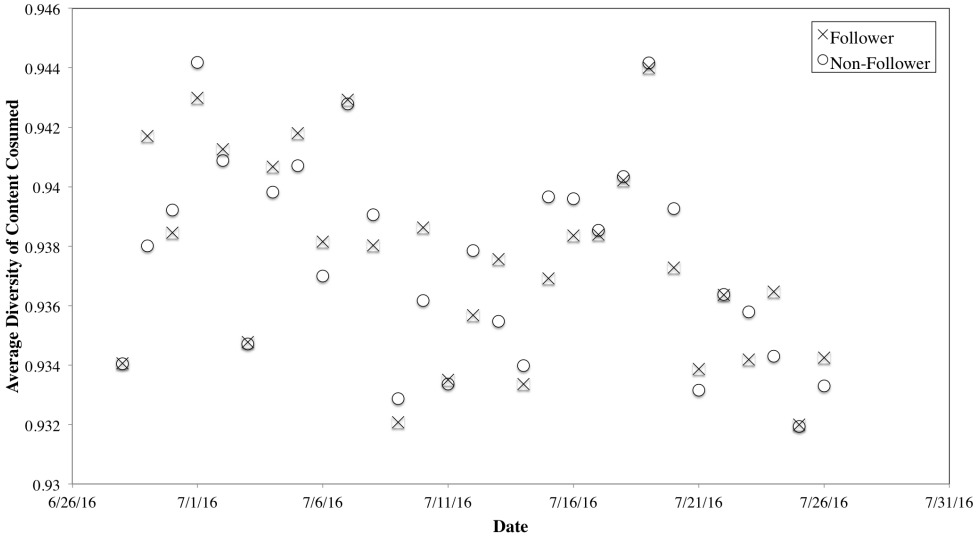


Table 5.2: Hypothesis tests for second research question.

Hypothesis	Accept/Reject
Users' initial diversity of content consumed directly correlates with expansion of diversity of content consumed.	reject
Users' click rate on recommendations directly correlates with expansion of diversity of content consumed.	reject
Users' activity level directly correlates with expansion of diversity of content consumed.	accept

tion in Table 5.2. We will discuss the results of both research questions' hypothesis tests in the following section.

5.3 Discussion

The first goal of our research was to investigate how serendipity in a recommendation system affects the filter bubble effect. Our results show that the serendipitous recommendations we served provided users with increasingly diverse recommendations, while the baseline recommendation diversity dipped at first but showed a slight upward trend later in the experiment. However, our results do not indicate that the recommendations had any effect on user consumption diversity. In our domain, the serendipitous algorithm was fulfilling its purpose of providing recommendations of increasing diversity, but the users were not changing their consumption patterns. This outcome suggests that personalization algorithms in our domain have less of an effect on filter bubbles than suggested by Pariser [45] and others. However, there are other publications with similar findings to ours.

We have two possible explanations for the mismatch between content recommended and content consumed. First, the serendipitous recommendations could not have been

as appealing to the users as the baseline. Our reason for suggesting this comes from analyzing the daily CTR on each recommender system. We calculated an average difference of 1.63% unique CTR and .14% raw CTR between the two recommender algorithms. Unique CTR only considers unique impressions and unique clicks, while raw CTR is total clicks divided by total impressions. A second explanation is that users are not interested in permanently expanding their content horizons even when nudged by recommenders. In either case, the problem lies more with the user actions than the algorithm.

To investigate our possible explanations, we took a closer look at the behavior of users in each group. We traced a few individual users from the serendipitous recommendation group who clicked on recommendations outside of their normal consumption pattern. What we observe in spot-checking is users clicking on likely serendipitous items but returning to their normal browsing on the following days. While non-exhaustive, our investigation shows that users might occasionally have a serendipitous discovery but it does not change their long-term browsing habits.

Our analysis that users are responsible for their own filter bubble more closely follow the claims of other filter bubble researchers discussed in Section 2.2. Bakshy et al. [6] also found the limited filter bubble effect they measured to be caused by user choice rather than the personalization algorithm. Other researchers who considered the filter bubble to be a user problem designed systems for users to understand their own filter bubbles to help them consciously break out of them [29, 56, 37]. These systems are different from serendipitous recommenders, because the user is made completely aware that the purpose of the system is to help them broaden their content horizons.

Our result of the baseline recommender kept a nearly steady or even slightly increasing diversity of content recommended is not consistent with past work. Nguyen et al.'s baseline recommender [38] was found to serve a narrowing set of content over the course of their experiment. The difference between our findings was surprising, especially considering that Nguyen et al. [38] also used an item-based collaborative filter. A possible explanation is the difference in experiment runtime. Our experiment ran for 1 month, while Nguyen et al.'s [38] research was done on a dataset covering 21 months. Perhaps we would have seen a narrowing effect if we had more time to run our experiment.

The second goal of our research was to investigate how different users were affected by the filter bubble. Our results showed that less active users experience a drop in diversity of content consumed, while active users' content consumed remains steadily more diverse. Unfortunately, we have not found past literature on user activity and the filter bubble to compare our results to. This results could be interpreted in two ways. First, one could see the active users as most likely to appreciate being shown content outside of their regular scope and actually consuming it. In contrast, one could argue that less active users are most in need of being shown content outside of their regular scope since the data suggests they discover it on their own less often.

The results of testing user CTR and user initial diversity were less conclusive. In both tests, each group showed very little change in diversity over time. Further, separating users by CTR showed absolutely no relationship with recommendation following and diversity of content consumed. This contrasts with Nguyen et al.'s work [38] who found users who followed recommendations to experience less of a filter bubble effect. Nguyen et al. [38] state that the difference was small yet still signifi-

cant. We attribute this difference to the difference in domains, which we touch upon in our concluding chapter.

As a final point of discussion, we would like to mention the possible threats to validity in our experiment. Two threats arise as part of the domain constraints are **selection of users** and **user attrition**. Using cookies to track users is not the most reliable method of monitoring users, and also required us to have a some requirements in choosing our users from the pool available. This means our selection was not entirely random, only users with reliable browser cookies were used. Further, some of our subjects' cookies were refreshed during the experiment period, causing us to loose track of them. We showed that the user dropoff was small in Section 4.2.2, but there was nevertheless a dropoff. These two threats to validity could be addressed in future research using a different domain.

Another point specific to software engineering research is the **use of open-source code** for research projects. As described in Chapter 4, we used a number of open-source libraries to implement our pipelines and recommendations algorithms. Wright et al. [63] point out in their 2010 publication that we must consider the external validity concerns that arise from using open source software projects. Popular software, such as MariaDB and Apache Spark, must be considered to be reliable for research. Smaller projects such as PredictionIO, which only just entered the incubation stage with Apache, are less widely used and could be a source of validation concern to some.

Chapter 6

Conclusions and Future Work

In wrapping up this thesis, we answer our research questions and discuss some directions for future work. Our **first research question** lead us to study how serendipity in a recommender system affects the filter bubble. As we have discussed, a filter bubble caused by a personalization algorithm is broken into two parts: consumption diversity and recommendation diversity. Our experiment shows that serendipity in a recommender system mitigates the issue of narrowing recommendation diversity, but not consumption diverity. Thus, our answer is that serendipity can give users the opportunity to have expanding diversity of recommendations served to them but this does not entirely break the filter bubble effect. In addition to recommender systems serving more diverse content, users must also choose to change their consumption patterns to fully break the filter bubble as we defined it.

With our answer in mind, an appropriate future direction for research develop is to further develop systems that help users understand, control, and overcome their own filter bubbles. Systems such as those described by Knijnenburg et al. [29] and Nagulendra et al. [37] give users the awareness and power to decide for themselves how their content is tailored. These solutions differ from serendipitous recommenders because they openly show the user their content scopes rather than algorithmically pushing them towards outside content. Of course, not all users are interested in overcoming their filter bubbles. This is where the next challenge that arises: finding users who are interested in expanding their content horizons. In answering our second research question, we begin to shed some light on this challenge.

Our **second research question** investigated how user characteristics might play a role in the filter bubble effect. The ability to find users most at risk of experiencing a filter bubble or users most likely to break their own filter bubble would help designers choose which users to target with horizon-expanding guidance. The most definitive result we achieved here was that highly active users experience less of a narrowing effect in the content they consume when compared to less active users. We found no connection with click through rate or initial content diversity and the filter bubble effect. Thus, the answer to our question is that, of the three user characteristics investigated in this thesis, user activity level is the only user characteristic connected to the filter bubble. As mentioned in our discussion, this might help future developers learn which users they would like to target in combating the filter bubble effect.

There are a few possible directions for future work that arise from our second

research question. Of course, there are countless other user characteristics that have yet to be investigated for connection with the filter bubble effect. Additionally, identifying which users experience more of a filter bubble is not the same as identifying users who might be interested in breaking out of a filter bubble. Identifying the latter users could help researchers more than identifying the former. Not all users are interested in breaking their filter bubbles, but our results suggest that more active users would be a good place to start.

Another point regarding our results and future work is domain influence. Users arrive at discussions on the Viva Forum in a variety of ways, including the existing panels on the home page or external sites such as Google Search results. The role of our recommender is small relative to domains where the recommendation engine is the primary means of user's content discovery. Further experiments are needed in domains where the recommender system plays a greater role in user content selections. If a serendipitous recommender were deployed in such a domain, perhaps the diversity of content consumed would be affected more.

One of the defining characteristics of the topics we have pursued is the difficulty in measuring them. The concepts of serendipity and the filter bubble effect are challenging to define and difficult to measure. Both serendipity and the filter bubble were defined for the purposes of our study, but no metric for these abstract topics is perfect. It would be beneficial to have more established definitions of these two concepts, allowing for easier comparison between studies.

Despite the challenges in measurement, we stand confidently by our findings. The filter bubble debate and the roles of recommendation algorithms and users certainly requires more study, but we hope to have made a small step forward in research with this thesis.

Bibliography

[1] Zeinab Abbassi, Sihem Amer-Yahia, Laks VS Lakshmanan, Sergei Vassilvitskii, and Cong Yu. Getting recommender systems to think outside the box. In *Proceedings of the third ACM conference on Recommender systems*, pages 285–288. ACM, 2009.

[2] Lada A Adamic and Natalie Glance. The political blogosphere and the 2004 us election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery*, pages 36–43. ACM, 2005.

[3] Gediminas Adomavicius and YoungOk Kwon. Improving aggregate recommendation diversity using ranking-based techniques. *IEEE Transactions on Knowledge and Data Engineering*, 24(5):896–911, 2012.

[4] Xavier Amatriain and Justin Basilico. Netflix recommendations: beyond the 5 stars (part 1). *Netflix Tech Blog*, 6, 2012.

[5] Chris Anderson. *The long tail: Why the future of business is selling more for less*. Hyperion, 2006.

[6] Eytan Bakshy, Solomon Messing, and Lada A Adamic. Exposure to ideologically diverse news and opinion on facebook. *Science*, 348(6239):1130–1132, 2015.

[7] Upasna Bhandari, Kazunari Sugiyama, Anindya Datta, and Rajni Jindal. Serendipitous recommendation for mobile apps using item-item similarity graph. In *Asia Information Retrieval Symposium*, pages 440–451. Springer, 2013.

[8] David M Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.

[9] Ilaria Bordino, Yelena Mejova, and Mounia Lalmas. Penguins in sweaters, or serendipitous entity search on user-generated content. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 109–118. ACM, 2013.

[10] Engin Bozdag and Jeroen van den Hoven. Breaking the filter bubble: democracy and design. *Ethics and Information Technology*, 17(4):249–265, 2015.

[11] Erik Brynjolfsson, Yu Jeffrey Hu, and Michael D Smith. From niches to riches: Anatomy of the long tail. *Sloan Management Review*, 47(4):67–71, 2006.

[12] José Campos and Antonio Dias de Figueiredo. Searching the unsearchable: Inducing serendipitous insights. In *Proceedings of the workshop program at the fourth international conference on case-based reasoning, ICCBR*, 2001.

[13] Michael D Conover, Bruno Gonçalves, Alessandro Flammini, and Filippo Menczer. Partisan asymmetries in online political activity. *EPJ Data Science*, 1(1):1, 2012.

[14] Marco de Gemmis, Pasquale Lops, Giovanni Semeraro, and Cataldo Musto. An investigation on the serendipity problem in recommender systems. *Information Processing & Management*, 51(5):695–717, 2015.

[15] Daniel Fleder and Kartik Hosanagar. Blockbuster culture's next rise or fall: The impact of recommender systems on sales diversity. *Management science*, 55(5):697–712, 2009.

[16] Daniel M Fleder and Kartik Hosanagar. Recommender systems and their impact on sales diversity. In *Proceedings of the 8th ACM conference on Electronic commerce*, pages 192–199. ACM, 2007.

[17] Andreas Forsblom, Petteri Nurmi, Pirkka Åman, and Lassi Liikkanen. Out of the bubble: Serendipitous even recommendations at an urban music festival. In *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*, pages 253–256. ACM, 2012.

[18] Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. Beyond accuracy: evaluating recommender systems by coverage and serendipity. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 257–260. ACM, 2010.

[19] Eric Gilbert, Tony Bergstrom, and Karrie Karahalios. Blogs are echo chambers: Blogs are echo chambers. In *System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on*, pages 1–10. IEEE, 2009.

[20] Eduardo Graells-Garrido, Mounia Lalmas, and Daniele Quercia. Data portraits: connecting people of opposing views. *arXiv preprint arXiv:1311.4658*, 2013.

[21] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

[22] Bryan Horling and Matthew Kulick. Personalized search for everyone. *The Official Google Blog*, pages 04–12, 2009.

[23] Kartik Hosanagar, Daniel Fleder, Dokyun Lee, and Andreas Buja. Will the global village fracture into tribes? recommender systems and their effects on consumer fragmentation. *Management Science*, 60(4):805–823, 2013.

[24] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272. Ieee, 2008.

[25] M Jenders, T Lindhauer, G Kasneci, R Krestel, and F Naumann. A serendipity model for news recommendation. In *Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz)*, pages 111–123. Springer, 2015.

[26] Junzo Kamahara, Tomofumi Asakawa, Shinji Shimojo, and Hideo Miyahara. A community-based recommendation system to reveal unexpected interests. In *11th international multimedia modelling conference*, pages 433–438. IEEE, 2005.

[27] Noriaki Kawamae. Serendipitous recommendations via innovators. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 218–225. ACM, 2010.

[28] Noriaki Kawamae, Hitoshi Sakano, and Takeshi Yamada. Personalized recommendation based on the personal innovator degree. In *Proceedings of the third ACM conference on Recommender systems*, pages 329–332. ACM, 2009.

[29] Bart P Knijnenburg, Saadhika Sivakumar, and Daricia Wilkinson. Recommender systems for self-actualization. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 11–14. ACM, 2016.

[30] Yehuda Koren, Robert Bell, Chris Volinsky, et al. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[31] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer, 2011.

[32] V Luckerson. Here's how facebook's news feed actually works. time, 2015.

[33] Valentina Maccatrozzo. Burst the filter bubble: using semantic web to enable serendipity. In *International Semantic Web Conference*, pages 391–398. Springer, 2012.

[34] CD Manning, P Raghavan, and H Schütze. H. introduction to information retrieval (vol. 1), 2008.

[35] Matt Marshall. Aggregate knowledge raises $5 m from kleiner, on a roll. *Venture Beat*, 2006.

[36] Tomoko Murakami, Koichiro Mori, and Ryohei Orihara. Metrics for evaluating the serendipity of recommendation lists. In *Annual Conference of the Japanese Society for Artificial Intelligence*, pages 40–46. Springer, 2007.

[37] Sayooran Nagulendra and Julita Vassileva. Understanding and controlling the filter bubble through interactive visualization: a user study. In *Proceedings of the 25th ACM conference on Hypertext and social media*, pages 107–115. ACM, 2014.

[38] Tien T Nguyen, Pik-Mai Hui, F Maxwell Harper, Loren Terveen, and Joseph A Konstan. Exploring the filter bubble: the effect of using recommender systems on content diversity. In *Proceedings of the 23rd international conference on World wide web*, pages 677–686. ACM, 2014.

[39] Dimitar Nikolov, Diego FM Oliveira, Alessandro Flammini, and Filippo Menczer. Measuring online social bubbles. *PeerJ Computer Science*, 1:e38, 2015.

[40] Derek O'Callaghan, Derek Greene, Maura Conway, Joe Carthy, and Pádraig Cunningham. The extreme right filter bubble. *arXiv preprint arXiv:1308.6149*, 2013.

[41] Gal Oestreicher-Singer and Arun Sundararajan. Recommendation networks and the long tail of electronic commerce. *Available at SSRN 1324064*, 2010.

[42] Kenta Oku and Fumio Hattori. Fusion-based recommender system for improving serendipity. In *Proceedings of the Workshop on Novelty and Diversity in Recommender Systems (DiveRS 2011), at the 5th ACM International Conference on Recommender Systems (RecSys 2011)*, page 19, 2011.

[43] Kenta Oku and Fumio Hattori. User evaluation of fusion-based approach for serendipity-oriented recommender system. In *Workshop on Recommendation Utility Evaluation: Beyond RMSE (RUE 2011)*, page 39, 2012.

[44] Kensuke Onuma, Hanghang Tong, and Christos Faloutsos. Tangent: a novel,'surprise me', recommendation algorithm. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 657–666. ACM, 2009.

[45] Eli Pariser. *The filter bubble: What the Internet is hiding from you*. Penguin UK, 2011.

[46] Yoon-Joo Park and Alexander Tuzhilin. The long tail of recommender systems and how to leverage it. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 11–18. ACM, 2008.

[47] Rebecca K Ratner, Barbara E Kahn, and Daniel Kahneman. Choosing less-preferred experiences for the sake of variety. *Journal of Consumer Research*, 26(1):1–15, 1999.

[48] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[49] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative filtering recommender systems. In *The adaptive web*, pages 291–324. Springer, 2007.

[50] Sylvain Senecal and Jacques Nantel. The influence of online product recommendations on consumers' online choices. *Journal of retailing*, 80(2):159–169, 2004.

[51] Guy Shani and Asela Gunawardana. Evaluating recommendation systems. In *Recommender systems handbook*, pages 257–297. Springer, 2011.

[52] Bracha Shapira, Francesco Ricci, Paul B Kantor, and Lior Rokach. Recommender systems handbook. 2011.

[53] N Singer. The trouble with the echo chamber online. *The New York Times*, 2011.

[54] Kazunari Sugiyama and Min-Yen Kan. Serendipitous recommendation for scholarly papers considering relations among researchers. In *Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries*, pages 307–310. ACM, 2011.

[55] Cass R Sunstein. *Republic. com 2.0*. Princeton University Press, 2009.

[56] Maria Taramigkou, Efthimios Bothos, Konstantinos Christidis, Dimitris Apostolou, and Gregoris Mentzas. Escape the bubble: Guided exploration of music preferences for serendipity and novelty. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 335–338. ACM, 2013.

[57] Clive Thompson. If you liked this, you're sure to love that. *The New York Times*, 21, 2008.

[58] Catherine Tucker and Juanjuan Zhang. Long tail or steep tail? a field investigation into how online popularity information affects the distribution of customer choices. 2007.

[59] Saúl Vargas and Pablo Castells. Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 109–116. ACM, 2011.

[60] Mian Wang, Takahiro Kawamura, Yuichi Sei, Hiroyuki Nakagawa, Yasuyuki Tahara, and Akihiko Ohsuga. Context-aware music recommendation with serendipity using semantic relations. In *Joint International Semantic Technology Conference*, pages 17–32. Springer, 2013.

[61] Jacob Weisberg. Bubble trouble: Is web personalization turning us into solipsistic twits. *Slate. com*, pages 10–06, 2011.

[62] Ian H Witten, Marco Gori, and Teresa Numerico. *Web dragons: Inside the myths of search engine technology*. Elsevier, 2010.

[63] Hyrum K Wright, Miryung Kim, and Dewayne E Perry. Validity concerns in software engineering research. In *Proceedings of the FSE/SDP workshop on Future of software engineering research*, pages 411–414. ACM, 2010.

[64] Hongzhi Yin, Bin Cui, Jing Li, Junjie Yao, and Chen Chen. Challenging the long tail recommendation. *Proceedings of the VLDB Endowment*, 5(9):896–907, 2012.

[65] Yuan Cao Zhang, Diarmuid Ó Séaghdha, Daniele Quercia, and Tamas Jambor. Auralist: introducing serendipity into music recommendation. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 13–22. ACM, 2012.

[66] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-scale parallel collaborative filtering for the netflix prize. In *International Conference on Algorithmic Applications in Management*, pages 337–348. Springer, 2008.