Norwegian University of Science and Technology

Master's thesis, spring 2014

User adaptation in an anonymous application setting

Author:
Jonas Myrlund

Supervisors:
Prof. Heri Ramampiaro
Prof. Helge Langseth

May 2014

Changelog

The latest changes are listed first.

2014-05-14 Cluster walkthrough and analysis. (HEAD, master)

2014-05-14 Started on clustering theory.

2014-05-13 Elaborated on chapter 3, in particular data cleaning and user modeling.

2014-05-13 Elaborated on generation and collection of event data.

2014-05-02 Elaborated on experiment design.

2014-04-14 Expanded on appear.in as a concept.

2014-03-31 Elaborated on evaluation chapter. Work in progress. (home/master)

2014-03-18 Tiny label fix. New version of PDF.

2014-03-18 Added condensed section on experimental setup.

2014-03-18 Added intro on similar domains and research.

2014-03-18 Expanded on how users are tracked in appear.in.

2014-03-14 Added changelog to compile pipeline and report.

2014-03-07 Some structural changes to report. Added ingestion pipeline figure.

2014-03-07 Updated motivation with a bit more on privacy concerns. Fixed chaptering.

2014-03-05 Fixed minor errors in report.

2014-03-05 Removed chapter 4 and moved higher ones to reflect.

2014-03-02 Updated report.

2014-02-17 Added preliminary version of the report source.

"I definitely need to find a timely quotation for my Master's thesis."

Jonas Myrlund, February 2014

NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Abstract

Faculty of Information Technology, Mathematics and Electrical Engineering
Department of Computer and Information Science

Master of Science in Computer Science

User adaptation in an anonymous application setting

by Jonas Myrlund

Identifying the different ways in which users use a product can be useful on several levels. In this project we will look at how identifying user classes allows for simple personalization of the product, and to what extent simple personalized treatments can alter user behavior.

It is often the case that some identified user classes are more desirable than others. Either because its associated users generate more revenue, use the product more, invite their friends, or similar. This project describes a system capable of not only identifying user classes, but more importantly a framework for identifying the most effective ways of driving users toward more desirable user classes.

More specifically, given a set of identified user classes and a set of predefined treatments, we want to find out how each treatment affects each user class. Although the project implementation will specifically target the video conferencing service appear.in, a major research question will be to what extent the results generalize.

We find that . . .

Acknowledgements

I would like to thank my supervisors, Heri Ramampiaro and Helge Langseth, for lots of good advice, while at the same time allowing me a great extent of freedom in planning and executing this project.

My supervisor at Telenor Digital AS – Svein Yngvar Willassen.

Contents

List of Figures

List of Tables

Abbreviations

Chapter 1

Introduction

1.1 Background and motivation

My motivation for doing a user adaptation project related to anonymous online video conferencing is comprised of two important factors. To understand how they are related and why they both are of equally big interest, some background on both the technological landscape of the web and the application case is needed.

For a long time, developing video conferencing services was an extremely challenging discipline, and as a result the market has consisted of a correspondingly low number of actors. However, this trend is currently in the process of being shaken and turned on its head with the introduction of HTML5; more specifically, with the introduction of the WebRTC[1] specification.

As the name implies, WebRTC handles real-time communication, but for our case, the important aspect of the technology is that it is able to do so peer-to-peer. Although it per design is a protocol for exchanging arbitrary data between peers, it is particularly geared toward multimedia streams. For instance, the traditionally cumbersome task of setting up a two-way audiovisual connection is now a matter of dropping around 40 lines of boilerplate Javascript into a web page[2].

[1]Web Real-Time Communication.
[2]For an excellent introduction, see: http://www.html5rocks.com/en/tutorials/webrtc/basics/.

Although the WebRTC specification is still officially a *working draft* in the W3C[3], several large browser vendors have already implemented it, and applications previously unseen on the web pop up every week.

1.1.1 Introducing appear.in

One of these applications is called appear.in, and it will serve as the main use case in this thesis. Like many other WebRTC applications, appear.in concerns itself with video conferencing. The idea is simple enough: a conversation happens between users who are in the same room at the same time. The central idea, though, is that the room is identified solely by the URL in use, and not in any way by the peers connecting. As an example, if any two people are visiting `https://appear.in/ntnu` at the same time, they will see and hear each other and can start chatting away without any further call setup or configuration.

This view of a conversation as not really being an enitity in its own right, but rather an *effect* of people being in the same room at the same time, breaks with the traditional model of audiovisual communication. Traditionally, talking to someone not present has been a process of one person *calling* the other one, with group conversations usually being nothing more than an extension of this concept. appear.in doesn't concern itself with distinguishing between callers and callees, has no simple concept of a "conversation", and generally does not enforce any particular way of using the service – apart from requiring the conversation venue to be identifiable by a string of characters.

1.1.2 Usage patterns

Until the arrival of WebRTC, this particular way of thinking about conversations for anything but textual conversations hasn't been a big thing. However, the simplicity of the room concept opens the service up for a wide variety of uses[4]. These wildly varied use cases are where the motivation for this project stems from:

1. If the users' behaviors are quantified, will any clear and distinct usage patterns emerge?

[3]The latest specification can be found here: `http://dev.w3.org/2011/webrtc/editor/webrtc.html`.

[4]In addition to traditional video calls, we've already seen it used for everything from virtual offices and team meeting rooms to baby monitoring and remote tutoring, just to name a few.

2. If so, can the different uses be better served by dynamically adapting the product to fit each of them?

1.1.3 Anonymity and privacy

appear.in is an anonymous communication service. No personal information is ever collected about the users, and not even IP-addresses or geolocational data is logged on an individual level. By tracking individual *browsers* using cookies, then logging behavioral events along with a cookie identifier, we can measure user behavior over time.

This all opens a wide series of questions bordering to sociological aspects of web usage:

1. To what extent can an anonymous web service be adapted? Is user behavior enough to provide a satisfactory personalized user experience?
2. Will a personalized user experience go against the users' expectations of appear.in as an anonymous web service?
3. How can we measure any of this?

1.1.3.1 Absence of identity

Several others have written about how privacy constraints impact personalized systems [1, 2]. In many ways, the very nature of anonymity is an extension of privacy. However, the absence of *identity* presents an entirely different challenge.

There are ways of coping with the absence of user identification. Kobsa, for instance, describes an approach that makes heavy use of pseudonyms designed around this problem [3]. However, appear.in is not only a pseudonymous service – it is a fully anonymous service; there are no user accounts, and there is no login. Tracking users, then, is obviously a bit of a struggle.

1.1.3.2 Tracking users

appear.in runs in the browser. The first time a user enters the site, a cookie is set with a random value uniquely identifying the browser over time. This value is sent with every event to the instrumentation service used for user analytics.

Unfortunately, using only cookies for identification has its clear downsides. Should the user clear the browser cache, switch browsers, use the browser "incognito"[5], or simply use multiple machines, then different user ids will be generated for each case.

Without tracking more user information – IPs and locations, for instance – there is no easy way of tying these user ids together[6], and to reason about them as a single user. However, collecting this information about users goes against appear.in's privacy policy, and it has been deemed more interesting to see what can be done without crossing this line.

1.1.4 Why personalize?

@TODO: Write out in full-text.

- Premise: Users use service in different ways, and want a simple UI targeted at their specific use case.
- Goal: Provide an interface which stimulates activation and retention.
- Result: Increase general user base, as well as their happiness (as measured by metrics such as recurring visits and time on site).

1.1.5 Visualization of clustering

@TODO: What is the state of the art regarding visualization of spacio-temporal clusters. (Dimensionality reduction etc.)

Package into clustering tool to improve business intelligence. Challenges from lack of demographics.

[5]Google lingo for private browsing, where previously set cookies are not available (among other things).

[6]Some suggestions on how to solve this is found in the suggestions for further work, in section 5.3.

1.2 Problem specification

Main research question: Can users of anonymous video conferencing services be clearly divided into user classes based on their behavior, and if so, to what effect can personalization improve their activity level?

1. Are users of video conferencing services such as appear.in clearly dividable into separate groups based only on their behavior within the service? Do these patterns reflect those seen elsewhere – in other types of internet services or even in real life?
2. Is it feasible to personalize treatments to these user classes? Does it stimulate users into becoming more active users?
   (a) Is this something these users want?
   (b) Do the inferred preferences of the detected user classes significantly differ from each other?
   (c) Can the personalized treatments be devised in such a way as to stimulate the moving of users in the direction of any desired user class?
3. How can a toolkit be devised to handle the following?
   (a) User classification based on behavior.
   (b) Product personalization based on a relevant user's class.
   (c) Tracking of each treatment's effect on each user class.
   (d) Prioritize using the most effective treatments without introducing statistical bias (see multi-armed bandit).
   (e) Allow product developers to easily access results to improve future feature prioritization.

1.3 Organization of the thesis

This paper is organized as follows.

Chapter 2 surveys relevant literature, similar applications, provides an in-depth analysis of the available data. Chapter 3 describes the system design and the reasoning behind central design choices. In chapter 4 we'll look at the execution results, and see how they evaluate. Chapter 5 summarizes the most important takeaways, and suggests further work.

Chapter 2

Survey

2.1 Similar problem domains

Although the case in question is a specific service, the techniques in use and the limitations needing to be dealt with should apply to many kinds of applications.

When demographics are absent and identity is highly unreliable, this will unevitably lead to some highly sparse usage data and quite a bit of noise in the user modeling data. Assuming that the usage patterns of the users actually differ, will these patterns be clearly and reliably identifiable? Will it be possible to base an adaptive personalization system on this kind of data? When using an anonymous system, will users welcome a personalized product, or will they regard this as breaking with their perceived concept of anonymity?

These are all questions that apply especially to appear.in, but that may also apply to many other anonymous, web-based systems.

2.2 Similar research

The approach taken to adaptive personalization is based heavily on the work by Vrieze in “Fundaments of Adaptive Personalization” [4].

2.2.1 Relevant literature

Teltzrow and Kobsa's work on privacy-driven personalization systems provides insight into a lot of the issues surrounding the lack of demographic information in personalization [1, 2]. However, the main focus of their work seems to be that users are more willing to provide demographic information if that information is not backtracable to themselves, through pseudonymous personalization. This matches the application in question quite poorly, as the goal is not to obtain demographics – rather to attempt to cope without it.

@TODO Explore and include literature on:

- Clustering and segmenting users, choice of algorithms etc.
- Visualizing and evaluating clusters.
- A/B testing, multivariate testing, multi-arm bandits.
- Motivations for adaptive personalization. Online business models?
- Anonymity: do users experience personalization as an overstep?

Chapter 3

Approach

3.1 System overview

The proposed system is constructed around the data flowing through it.

The system takes the data stepwise through an ingestion pipeline, importing, filtering and cleaning it, before churning it into user models. This particular part of the system is elaborated on further in section 3.2.

Once user models are in place, the system generates user segments. These are stored in a database for future use. There are several viable approaches to the segmentation task. These are discussed in more detail in section **??**.

As it turns out, when designing completely autonomous adaptative systems, we need more data than user segments to effectively drive the adaptive component. When considering whether to enable a feature or an interface switch, we need to know whether doing so will be advantageous to the user segment in question. This is discussed in section 3.6.

3.2 Data ingestion and preprocessing

Before the interesting parts of the system can start doing their work, the data needs to be transformed from *a series of chronological raw events* to *a set of user models.*

The amount of data can be arbitrarily sizable, and will grow linearly with user activity. The system architecture has been designed to be able to cope with this; its functional and data-driven nature should be easily adaptable to hugely scalable programming paradigms like MapReduce.

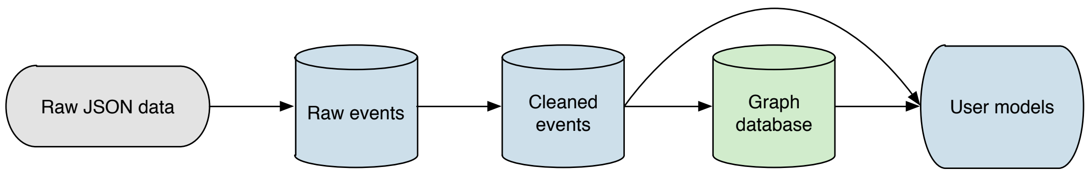


Figure 3.1: The ingestion pipeline broken into 4 steps. The color of each node indicates means of storage: *Blue* indicates a RDBMS, *green* indicates a graph database, whereas *gray* is used to indicate flat file storage.

3.2.1 Generating the raw data

The system input is a chronological series of raw events sent from the production system.

These are instrumented via an external analysis service called KISSmetrics[1]. It is a user analysis system designed around tracking individual users' behavior. It works by calling their REST API with the following data:

1. person identifier
2. event name
3. user properties

The consistency of the personal identifier has already been discussed extensively in the introductory chapters, especially section 1.1.3, but a short technical introduction to the actual production system is in order.

3.2.2 Event instrumentation in KISSmetrics

To enable effective utilization of the KISSmetrics instrumentation functionality, they supply a client library for the purpose. This client library handles a few central things for us:

[1] https://www.kissmetrics.com/

1. Person identity storage and loading over subsequent page loads.
2. The low-level instrumentation of events.
3. Simple A/B testing facilities.

When the KISSmetrics client library is loaded, the person identity is automatically either retrieved from the browser cookies, or generated.

The identity of a person is a unique randomly generated string, which serves no other purpose than to track the identity of the browser over time. No personal data is stored, nor is it available to us.

Whenever something "interesting" happens, an event is sent to the KISSmetrics instrumentation service. An "interesting" event is anything that tells us about how the users use the service, both in terms of general activity and in terms of feature adoption. Every event is tagged with the person identity, as well as an event name and a timestamp.

The KISSmetrics service provides several analytical tools to dig into this data, thereamongst funnel reports and cohort reports – as depicted in figure 3.2 and figure 3.3.

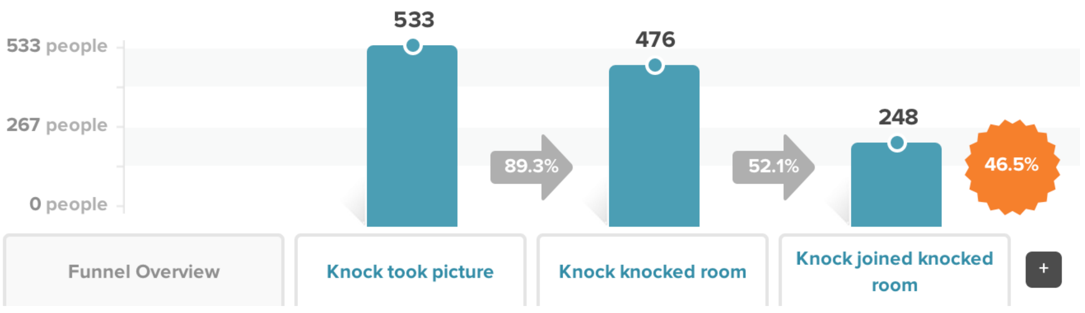
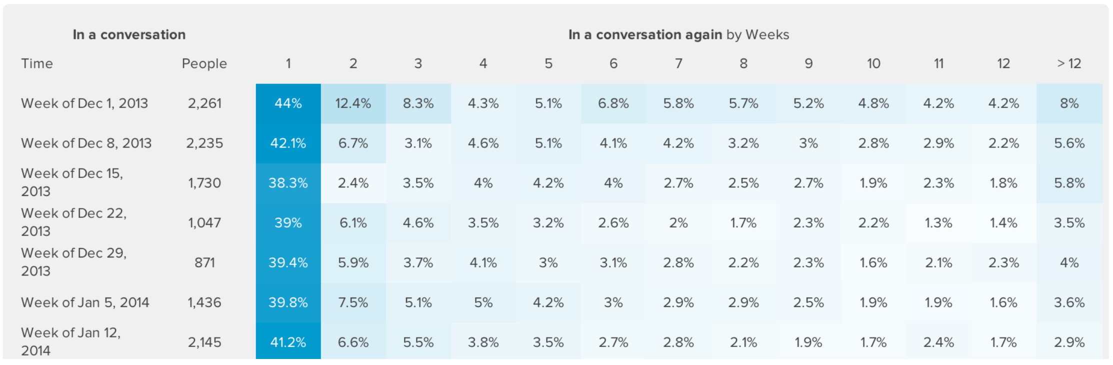


FIGURE 3.2: Example of a simple funnel report.



In a conversation		In a conversation again by Weeks												
Time	People	1	2	3	4	5	6	7	8	9	10	11	12	> 12
Week of Dec 1, 2013	2,261	44%	12.4%	8.3%	4.3%	5.1%	6.8%	5.8%	5.7%	5.2%	4.8%	4.2%	4.2%	8%
Week of Dec 8, 2013	2,235	42.1%	6.7%	3.1%	4.6%	5.1%	4.1%	4.2%	3.2%	3%	2.8%	2.9%	2.2%	5.6%
Week of Dec 15, 2013	1,730	38.3%	2.4%	3.5%	4%	4.2%	4%	2.7%	2.5%	2.7%	1.9%	2.3%	1.8%	5.8%
Week of Dec 22, 2013	1,047	39%	6.1%	4.6%	3.5%	3.2%	2.6%	2%	1.7%	2.3%	2.2%	1.3%	1.4%	3.5%
Week of Dec 29, 2013	871	39.4%	5.9%	3.7%	4.1%	3%	3.1%	2.8%	2.2%	2.3%	1.6%	2.1%	2.3%	4%
Week of Jan 5, 2014	1,436	39.8%	7.5%	5.1%	5%	4.2%	3%	2.9%	2.9%	2.5%	1.9%	1.9%	1.6%	3.6%
Week of Jan 12, 2014	2,145	41.2%	6.6%	5.5%	3.8%	3.5%	2.7%	2.8%	2.1%	1.9%	1.7%	2.4%	1.7%	2.9%

FIGURE 3.3: Example of a cohort report.

3.2.3 Cleaning the data

KISSmetrics has the ability to export raw event data to data files. This can be used to power completely customized analyses, as will be needed for our particular task.

After the raw data has been aquired, it will need to be cleaned. This is a simple process of churning through each line of each unprocessed data file, parsing and splitting its contents into appropriate data fields, and inserting it into a database.

To facilitate the compilation of network-related user model features, conversation data was loaded into a graph database to enable querying of network structures. In the RDBMS handling the other data, this presented itself as a computationally unfeasable task.

3.3 User modeling

To find clusters of users, we first and foremost need to quantify them. More specifically, we want to represent each user as a feature vector in an n-dimensional space.

3.3.1 Feature selection

Given the types of events being collected, we landed on compiling the following features:

1. First degree conversation partners
2. Second degree conversation partners
3. Inviter
4. Invitee
5. Conversations
6. Rooms used
7. Rooms claimed
8. Roomnames generated

9. Chat message sent

To generate user models, the stream of event data needs to be mapped to its associated users for aggregation. This is a classic MapReduce task [5] easily solved using simple functional techniques, and did not present itself as a particularly interesting problem, albeit a computationally heavy one.

To summarize, the transformation in this step started with data in the form of (3.1).

$$\langle \text{event}, \text{timestamp}, \text{person}, \text{metadata} \rangle \tag{3.1}$$

And ended with data in the form of (3.2).

$$\langle \text{person}, \text{feature}, \text{value} \rangle \tag{3.2}$$

3.4 Clustering the users

We assume the following hypothesis holds, as a basis for the clustering approach to the user adaptation problem:

Hypothesis 1. The more similarly two people use a service, the more likely they are to respond similarly to it changing.

Thus, we want to segment the users in the following way: users in each segment should be as similar as possible, and as dissimilar those in other segments as possible. This scheme fits well with the two criterias for selecting an optimal clustering scheme, as described by Berry and Linoff [6].

3.4.1 Choice of algorithms

Three algorithms were implemented and experimented with: DBSCAN, mean-shift, and k-means. Of these, the k-means clearly proved itself as the most effective one, and as it also managed to produce adequate and meaningful results, it was chosen as the principal algorithm.

3.4.1.1 The k-means clustering algorithm

The k-means clustering algorithm is arguably one of the most intuitive clustering algorithms.

It takes as input a preset number of clusters, k, a similarity measure function, and a set of data vectors. Initially, it randomly chooses k data vectors as centroids as a starting point for the process, before repeatedly performing a two-pass operation adjusting the centroids until convergence.

Figure 3.4 shows a simple python-esque implementation of the algorithm.

```python
def kmeans(K, distance_fn, vectors):
    N = len(vectors)
    cluster_members = {}
    memberships = {}

    # Initially set centroids to random data vectors
    centroids = [vectors[randint(N)] for _ in xrange(K)]

    while True:

        # Assign each data vector to the closest cluster centroid
        for vector in vectors:
            k = argmin(K, lambda k: distance_fn(centroid[k], vector))
            if not memberships[vector] == k:
                cluster_members[memberships[vector]].remove(vector)
                cluster_members[k].append(vector)
                memberships[vector] = k

        # Set each centroid to the mean of its members
        previous_centroids = centroids
        for k in xrange(K):
            centroids[k] = mean_vector(cluster_members[k])

        # Stop computing if we've achieved convergence
        change = 0.0
        for k in xrange(K):
            change += distance_fn(previous_centroids[k], centroids[k])
        if change < CONVERGENCE_THRESHOLD:
            break
```

FIGURE 3.4: The k-means algorithm

3.4.2 Evaluating cluster quality

@TODO: Davies-Bouldin index etc.

3.4.3 Normalizing axes

3.4.4 Storing clustering results

3.5 Identifying adaptable product features

Ideally at this point, we have identified several significant clusters of users. Now we want to adapt the product to better suit each user, based on the cluster he or she belongs to.

Given a set of candidate product alterations A, we want to give each user the combination of these that maximizes some performance measure P. However, we arguably have no predictive bias to start us off on solving this.

The approach taken in this implementation is a simple one, which relies on conducting a single A/B test up front for each candidate product alteration. The important part during this phase is to log each selected alteration variant along with the same person identifier used for other event instrumentation, as discussed in section 3.2.1.

This approach enables us to see up-front whether there are any significant variances in feature adoption between the clusters. If there are, we can adapt the service by selecting the more successful variation of the feature for all users in the relevant clusters. If not, we have a feature that affects all users equally – at least with regard to cluster granularity.

3.5.1 Evaluation metrics

When considering which variation of a product feature to prefer, we need to be able to measure their relative success rates.

As for any user-facing product, the main objective is achieving user happiness. However, user happiness is hard to objectively define. For a service like appear.in, though, activity level should serve as a pretty clear indicator of user happiness – unhappy users have more than enough alternative applications that cater to their needs.

Simply put, we basically assume that the following hypothesis holds:

Hypothesis 2. Happy users use the service more than unhappy users.

Furthermore, we seldom need a measure of *user happiness* to determine the relative success of variations of a product feature. Some parts of the product have clearly defined goals themselves.

The perhaps most obvious example of this is the landing page, whose main objective is to get people to try out the product. We can define the performance of the landing page in terms of the percentage of users that continue on to try out the product.

3.6 Adaptation component

3.6.1 Applying the personalized feature set

Description of the FlagService model.

3.6.2 Visualizing effects

3.6.3 Differentiating product features

3.7 Evolving the user models

3.7.1 Multi-arm bandits

3.7.2 Tracking individual treatment

3.8 Visualization requirements (?)

Chapter 4

Evaluation

4.1 Description of the evaluated dataset

4.2 Prerequisites

For any of this user adaptation to have any actual use, we will need to find out whether there is actually significant variation in feature adoption between clusters; it matters little whether users exhibit differing behavior with regard to existing functionality, if this behavior yields no basis for predicting feature adoption in the future. We need an inductive bias.

An easy way of discovering whether this inductive bias is present in the user base is to simply log whether there is significant variance across the user groups in their users' adoption of new features. Thus, this question of whether the predictive bias actually exists will be included as a central part of the actual experiment itself, and be thoroughly discussed through the next sections.

4.3 Experimental setup

Condensed version.

The main experiment determines whether an alteration of the service affects different kinds of users differently. More specifically, do users in clusters $C_1, \ldots, C_n$ employ function f in significantly differing ways? We find out without touching a large percentage of the users, and can use the results to individually enable features for users that

We break the process into three steps, each elaborated on in 4.3.2, 4.3.1, and 4.5.1:

1. A/B test feature f on a percentage of all users.
2. Separate users into clusters $C_1, \ldots, C_n$.
3. Analyze results: did the test results vary significantly between clusters?

If the final answer is "yes", we can store our results and use them as a basis for adaptating the interface for each user.

4.3.1 A/B testing features

@TODO: Brief description of how to fit the actual testing regime into the production system.

4.3.2 User clustering

@TODO: Describe methods and algorithms used. What kind of results are expected?

4.4 Initial clustering results

The data used in this analysis stems from February 2014.

The following clustering results were found by choosing the best of 5 k-means runs, "best" being defined by their Davies-Bouldin indices (see section 3.4.2 for a brief description of this evaluation metric). This process was performed for k parameter values from 3 to 10, where $K = 8$ yielded the best result. The 2 smallest resulting clusters were omitted in the analysis due to their relatively insignificant sizes.

Data from January and March yield more or less the same results, although they are a bit less clear. This could be due to media events and holidays generating more skewed data than usual.

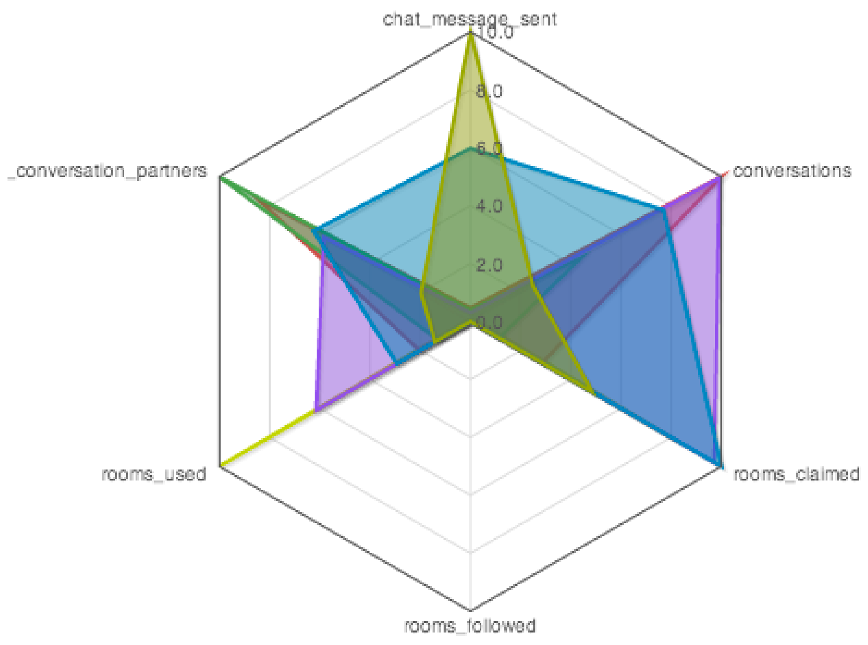


FIGURE 4.1: Radar chart comparing clusters generated from data for February 2014.

The radar chart in figure 4.1 shows the centroids of 6 large clusters C relative to each other.

Each dimension's centroid values $\mu_i \in \mu$ have been scaled by a factor of $\frac{10}{\max_{c \in C} c_i}$, to fit nicely inside the chart.

The features used in this particular clustering run are (going clockwise around the chart):

1. chat messages sent
2. conversations (2+ persons present in room)
3. rooms claimed
4. rooms followed
5. unique rooms used
6. conversation network size (ie. number of unique other users within 2 degrees of conversation separation)

Most of these features are quantified by counting the number of relevant events logged for each user.

4.4.1 The typical clusters

To know what types of users we are dealing with through the next chapters, let's walk through the clusters discovered in the February dataset, as described above.

4.4.1.1 Cluster 1: Front page hits

The centroid for this cluster is quite simply $\mu_1 = \langle 0, 0, 0, 0, 0, 0 \rangle$.

Persona 1. The user has not tried the actual service – most likely hitting the front page and either not using a compatible device/browser, or not finding it interesting enough to try out.

4.4.1.2 Cluster 2: Trying out the service alone

As shown in figure 4.2, the users in this cluster score close to 0 on every feature except the number of rooms used – most notably, the number of conversations.

Persona 2. The user has tried out the service, but not ever conversed with another user.

4.4.1.3 Cluster 3: Simple users with small networks

Users in cluster 3 (see figure 4.3) don't use any of the more advanced features of the service, like chatting, claiming or following a room, but on average they have taken part in just over 5 conversations.

Persona 3. A returning user, only utilizing the bare functionality, conversing only with a very limited group of people.

4.4.1.4 Cluster 4: Simple users with large networks

The users in cluster 4 (see figure 4.4) are very much like the ones in cluster 3, but use the service slightly more, and are part of much larger networks.

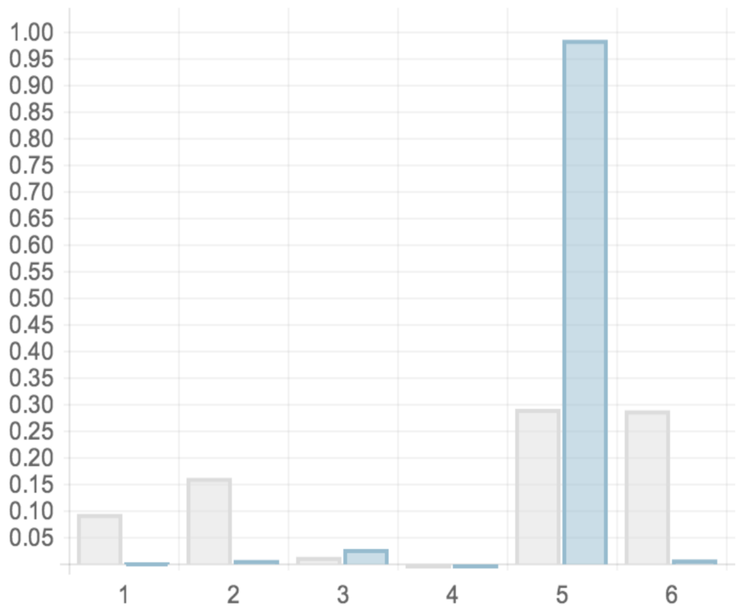


FIGURE 4.2: Cluster 2: Users trying out the service alone

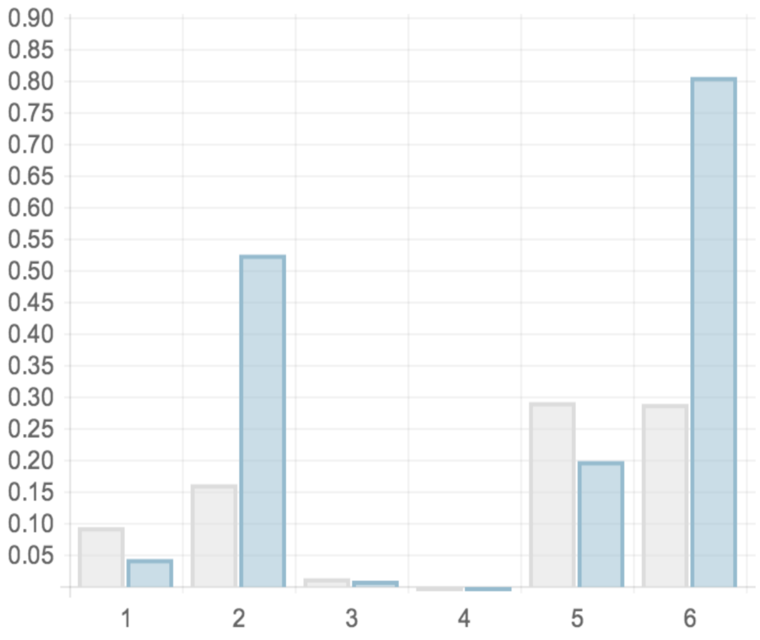


FIGURE 4.3: Cluster 3: Simple users with small networks

Persona 4. A returning user, only utilizing the bare functionality, part of a large group of people using the service.

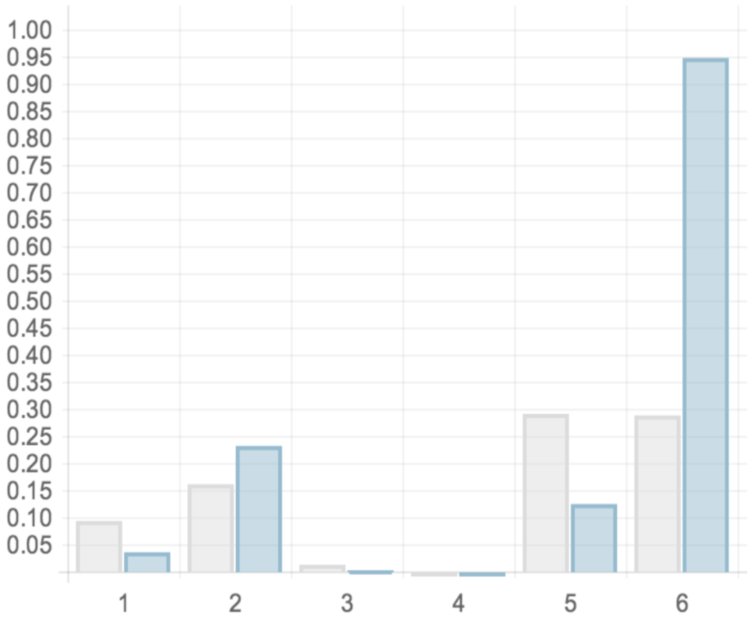


FIGURE 4.4: Cluster 4: Simple users with large networks

4.4.1.5 Cluster 5: Incognito users

To track users over time, cookies are required. Whenever clearing the browser cache, or when browsing in "incognito mode"[1], the service will not be able to tie together user sessions. These perceived one-off users should end up in this cluster, close to the pattern shown in figure 4.5.

Persona 5. A user browsing in incognito mode, or who clears the browser cache regularly.

4.5 Adapting the user interface

The adaptation component consists of two important steps.

1. Determine which type of user (ie. *cluster*) will have the most to gain from having feature f enabled.

2. Tagging the users with cluster and enabling features for

[1]Browsing without storing any data, including cookies.

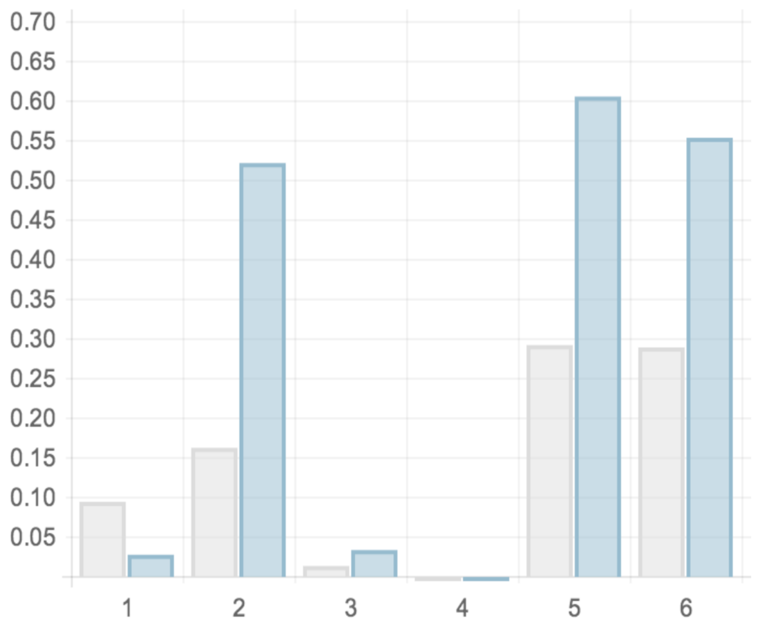


FIGURE 4.5: Cluster 5: Simple users with small networks

4.5.1 Evaluating the results

Seeing as the application does not yield any direct and real payoff, we will evaluate the performance of each variation relatively, using some general key metrics. This approach is described in [7].

For appear.in, the most important key metrics are "time on site" and "number of visits in the last n days". These are combined in providing a relative success metric for each variation.

@TODO: Verify key metrics.

4.6 Results

4.7 Discussion

Chapter 5

Conclusion

5.1 Summary

5.2 Generalizing the system

5.3 Suggestions for further work

Appendix A

Evaluation Results

Bibliography

[1] Maximilian Teltzrow and Alfred Kobsa. Impacts of user privacy preferences on personalized systems. *…personalized user experiences in eCommerce*, (0308277), 2004. URL http://link.springer.com/chapter/10.1007/1-4020-2148-8_17.

[2] Alfred Kobsa. Privacy-Enhanced Web Personalization. pages 628–670, 2007.

[3] Alfred Kobsa and Jörg Schreck. Privacy through pseudonymity in user-adaptive systems. *ACM Transactions on Internet Technology*, 3(2):149–183, May 2003. ISSN 15335399. doi: 10.1145/767193.767196. URL http://portal.acm.org/citation.cfm?doid=767193.767196.

[4] Paul De Vrieze. *Fundaments of Adaptive Personalisation*. ISBN 9789090211138.

[5] Jeffrey Dean and Sanjay Ghemawat. MapReduce : Simplified Data Processing on Large Clusters. *Communications of the ACM*, 51(1):1–13, 2008. ISSN 00010782. doi: 10.1145/1327452.1327492. URL http://portal.acm.org/citation.cfm?id=1327492.

[6] Michael J. Berry and Gordon Linoff. *Data Mining Techniques: For Marketing, Sales, and Customer Support*. John Wiley & Sons, Inc., New York, NY, USA, 1997. ISBN 0471179809.

[7] Yisong Yue, Josef Broder, Robert Kleinberg, and Thorsten Joachims. The K-armed dueling bandits problem. *Journal of Computer and System Sciences*, 78(5):1538–1556, September 2012. ISSN 00220000. doi: 10.1016/j.jcss.2011.12.028. URL http://linkinghub.elsevier.com/retrieve/pii/S0022000012000281.