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.

2014-05-28 Expanded survey chapter greatly. (HEAD, origin/master, origin/HEAD, master)

2014-05-28 New reference style.

2014-05-28 Minor fixes.

2014-05-28 Rewrite of abstract, problem definition and research questions.

2014-05-28 Started namespacing labels.

2014-05-28 Started on historic and contextual literature survey in chapter 2.

2014-05-28 Expanded chapter 1.

2014-05-26 Rewrote central parts of introductory chapter. Moved out survey-y stuff to ch2.

2014-05-25 Figures into subfigures in chapter 4.

2014-05-24 Reworked structure of chapter 1. Introduction/background/motivation rewrite.

2014-05-14 Cluster walkthrough and analysis.

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.

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

The goal of the project in this thesis is to explore the viability of a new approach to user adaptation, where the requirements of the application are significantly more constraining than in most cases seen in previous academic work.

In this project we will describe a system for rolling out product features incrementally in an optimal way, based on feature adoption statistics within user segments. In other words, the described system allows for simple personalization of the product while experimenting with different feature variations.

Identifying the different ways in which users use a product can be useful on several levels, and it is often the case that some are more desirable than others. This can be due to its associated users generating more revenue, using the product more, inviting their friends, or similar. This project describes a system capable of not only identifying user classes based on user behavior, but more importantly: a framework for identifying the most effective ways of adapting the product to these user class segments, in effect driving users in a desirable direction.

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 there are indeed clear differences in feature adoption across the identified user segments. However, due to uncertainty caused by domain constraints, it is hard to tell to what extent the results generalize.

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

The goal of the project in this thesis is to explore the viability of a new approach to user adaptation, where the requirements of the application are significantly more constraining than in most cases seen in previous academic work.

The application case is an anonymous, web-based video conferencing service called `appear.in`. The most important part of that description is the term "anonymous", as we not only need to deal with a complete lack of demographic information about the users – we do not even have the ability to consistently identify them. Luckily, there are a few workarounds to the issue of identification that make the problem at least partly tractable.

Previous work in the area of user adaptation has often adhered to the concept of recommending some kind of resource – often being reducible to the task of finding relevant items to sell, or interesting pages in a website hierarchy. The goal of this project, however, is not to *find good things*, it is to *optimize the user experience of an application*. More specifically, the goal is to be able to predict which version of the application will lead to better performance for each user, down to the individual feature level.

Let us start off with some background on why I believe this is an especially exciting time for user adaptation on the web, before section 1.2 introduces the specific application case of appear.in. Finally, section 1.3 will provide an introduction to the realm of user adaptation.

1.1 Background and motivation

The web is an interesting venue in which to explore new ways of user adaptation and personalization. Even today, when we are able to build almost every kind of application right in the web browser, a fundamental concept remains: there is still a server serving the application code every single time a user opens the site[1].

1.1.1 The modern web

This is the modern web: a place where applications are nearing the power of traditional desktop applications, but with the potential of being continuously tailored to suit each user individually.

When we say that web applications are nearing the power of traditional desktop applications, we are of course talking about the introduction of HTML5. HTML was primarily designed as a language for semantically describing documents, and little more. The last decade, however, the concept of web applications has thoroughly established itself, while lacking clear standardization efforts from the W3C. The HTML5 specification is an attempt to remedy this [1], by providing standards and guidelines for the browser vendors on how to implement a wide range of common APIs.

In practice, these APIs lets websites do things like:

- play audio and video[2]
- generate graphics[3]
- access your webcam and microphone[4]
- handle and manipulate arbitrary files[5]
- send and recieve data over full-duplex socket connections[6]

[1] Caching aside, of course.
[2] http://dev.w3.org/html5/spec-author-view/video.html
[3] http://www.w3.org/TR/2dcontext/
[4] http://www.w3.org/TR/mediacapture-streams/
[5] http://www.w3.org/TR/FileAPI/
[6] http://www.w3.org/TR/websockets/

...as well as a wide range of other things.

In general, HTML5 enables web applications to do most of the things one would need plugins or native applications for just a few years ago.

One of these new API specifications is called WebRTC[7], and it is the last piece of the API puzzle enabling applications like appear.in.

1.2 The application case

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 specification.

1.2.1 The enabling piece of technology: WebRTC

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[8].

Although the WebRTC specification at the time of writing still officially is a *working draft* in the W3C[9], most of the large browser vendors have already implemented it. Consequentially, a plethora of applications leveraging this technology are already available, with ever more being launched every month.

[7] Web Real-Time Communication.
[8] For an excellent introduction, see: http://www.html5rocks.com/en/tutorials/webrtc/basics/.
[9] The latest specification can be found here: http://dev.w3.org/2011/webrtc/editor/webrtc.html.

1.2.2 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 *conversation* 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.

For both screenshots, as well as a thorough breakdown of the application, please see section 2.1.1.

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.

1.2.3 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[10]. 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?

[10] 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?

1.2.4 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, however, we can track users – or more precisely, browsers – over time.

By logging various events that we deem interesting along with a cookie value identifying the browser, we can reconstruct user sessions, and connect them to the application users pseudonomously. By "pseudonomously" it is meant that we identify the users solely by a random string set in their browser.

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?
2. Is user behavior enough to provide a satisfactory personalized user experience?

These are issues which will be revisited throughout the thesis.

1.2.4.1 Absence of identity

Several others have written about how privacy constraints impact personalized systems [2, 3]. 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 [4]. 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.

1.2.4.2 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"[11], 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[12], 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.

1.3 User adaptation

Before answering the question of *how* to do user adaptation, let's look into the more fundamental issue of *why*. Why would we want to adapt a product to its users?

There can be several reasons. Dijkstra once described a way in which humans and computers differ with a short anecdote involving piano playing and his mother [5]:

> To end up my talk I would like to tell you a small story, that taught me the absolute mystery of human communication. I once went to the piano with the intention to play a Mozart sonata, but at the keyboard I suddenly changed my mind and started playing Schubert instead. After the first few bars my surprised mother interrupted me with "I thought you were going to play Mozart!". She was reading and had only seen me going to the piano through the corner of her eye. It then transpired that, whenever I went to the piano, she always knew what I was going to play! How? Well, she knew me for seventeen years, that is the only explanation you are going to get.

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

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

Humans constantly monitor the world around them, and model the objects and people in it. The kind of implicit modeling and predictive behavior is something computers generally have a hard time dealing with.

In addition to this, the digital world around us is becoming more and more complex. This complexity will at some point overwhelm users. Adapting to the user's expectations may help in this [6].

For a survey of the field of user adaptation, see section 2.2.

1.3.1 A new context

The application case of appear.in presents a few novel qualities for an adaptive system to deal with. Firstly, we want to do feature-level user adaptation of a web application, and secondly, we want to do so with extremely challenging privacy constraints.

The first point, what being an *application* running on the web entails for the relevance of previous research, is discussed in depth in section **??**.

The second point, the impact of tight privacy constraints, requires some discussion before we take on the concrete problem specification of this thesis.

As we shall see throughout chapter 2, most of the surveyed adaptation systems have a notion of *users*, who can be tracked over time. Furthermore, most traditional user modeling systems even assume either the availability of demographic information or some notion of *consumption of resources*, which in turn both can aid in constructing user models.

We, however, do not enjoy such luxuries. A large part of this thesis will look at how far we can get with only behavioral data, when even identity is at best an unstable notion.

Section 2.2.5 will take a closer look at previous research on the notion of "privacy versus personalization".

1.4 Problem specification

An essential part of the research will consist of determining whether anonymous users, like the ones described for the appear.in application, will be clearly separable into clusters based purely on their behavior – the only data available about them.

Given a set of identified user classes and a set of product variations, we want to find out how each variation affects each user class. When launching a new feature, for instance, will it be adopted differently by different classes of users?

To find out, we perform the following steps:

1. Separate users into user classes using segmentation techniques.
2. Perform A/B tests of applicable feature variants, independently of user classes.
3. Look for significant variation between user classes given their designated variation, as measured by some performance measure.

If significant variation exists for a feature's variations, we can consistently select the highest-scoring feature variation for the members of the various user classes. The reasoning behind this leap can be found in section 3.4.

The A/B testing regime proposed in this work has the advantage of not requiring any sort of reasoning about the actual users in each clusters. By all means, this may well be interesting from a business intelligence perspective, but it will not be necessary to effectively roll out an optimized feature set for each user class.

This project describes a system capable of not only identifying user classes based on user behavior, but more importantly: a framework for identifying the most effective ways of adapting the product to these user classes, in effect driving users in a desirable direction.

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.

1.4.1 Research questions

With the context outlined above, we formulate the following two main research questions:

1. Is it possible to clearly divide users of simple, anonymous web applications into user classes based solely on their behavior?
2. To what extent can such an application be adapted to better suit their way of using it?

It is clear that the second research question strongly depends on the first one, and consequently the thesis is organized in a similarly stepwise manner.

1.5 Organization of the thesis

This paper is organized as follows.

Chapter 2 surveys similar research, similar applications, and provides an in-depth analysis of the available data. Chapter 3 describes the system design and the reasoning behind central design choices, choices of algorithms et cetera. In chapter 4 we'll analyze the discovered user classes to help answer the first research question, and see how they differ in their adoption of two central application features. 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.

Firstly, let's pick apart the workings of appear.in, to thoroughly understand what kind of an application we are dealing with – a prerequisite for understanding the generated data and the applicability of the results herein for the general case.

2.1.1 The inner workings of appear.in

As illustrated in figure 2.1, the appear.in architecture is quite simple. It is built on a simple peer-to-peer (P2P) architecture, with signaling done through a centralized server endpoint.

The instrumentation service has been included, simply to illustrate the fact that the data we have available is generated and sent directly from the clients.

By "signaling" in the context of appear.in, we mean everything not directly related to the media streams between the peers. This includes:

- Managing which peers are in which room.
- Setting up new peer connections when a client joins a room.

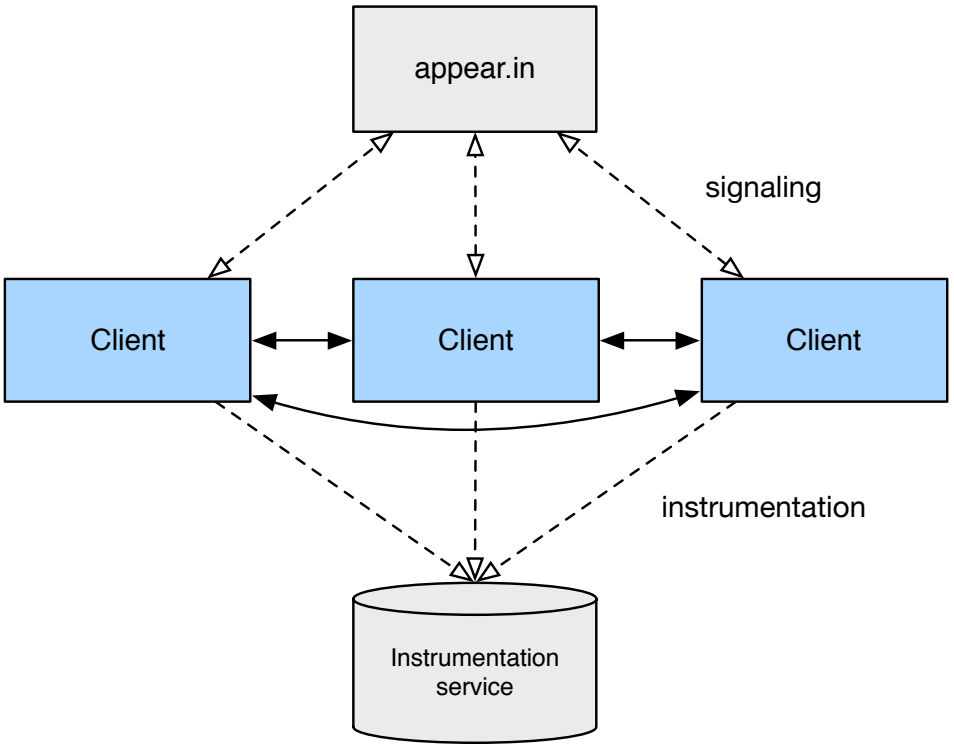


FIGURE 2.1: The appear.in architecture, illustrated for a conversation between 3 peers. The black arrows indicate media data flow, and the dashed arrows indicate signaling data and instrumentation, as indicated.

- Tearing down peer connections when a client exits a room.
- Distributing various metadata that needs to be in sync across peers.

The user interface is also quite simple. It consists of a landing page (a screenshot of the current version can be seen in figure 2.2), and a "room page" (see figure 2.3). For the sake of simplicity, let's go through them separately.

2.1.1.1 The landing page

The landing page's objective, as for most landing pages, is two-fold: to *evoke interest*, and to *activate the user*. Although we cannot directly measure them, we can indirectly measure the degree of interest and the activation rate in two ways:

1. The ratio of users going from the landing page to a room (interest).
2. The ratio of users going from the landing page to a room to a conversation (activation).

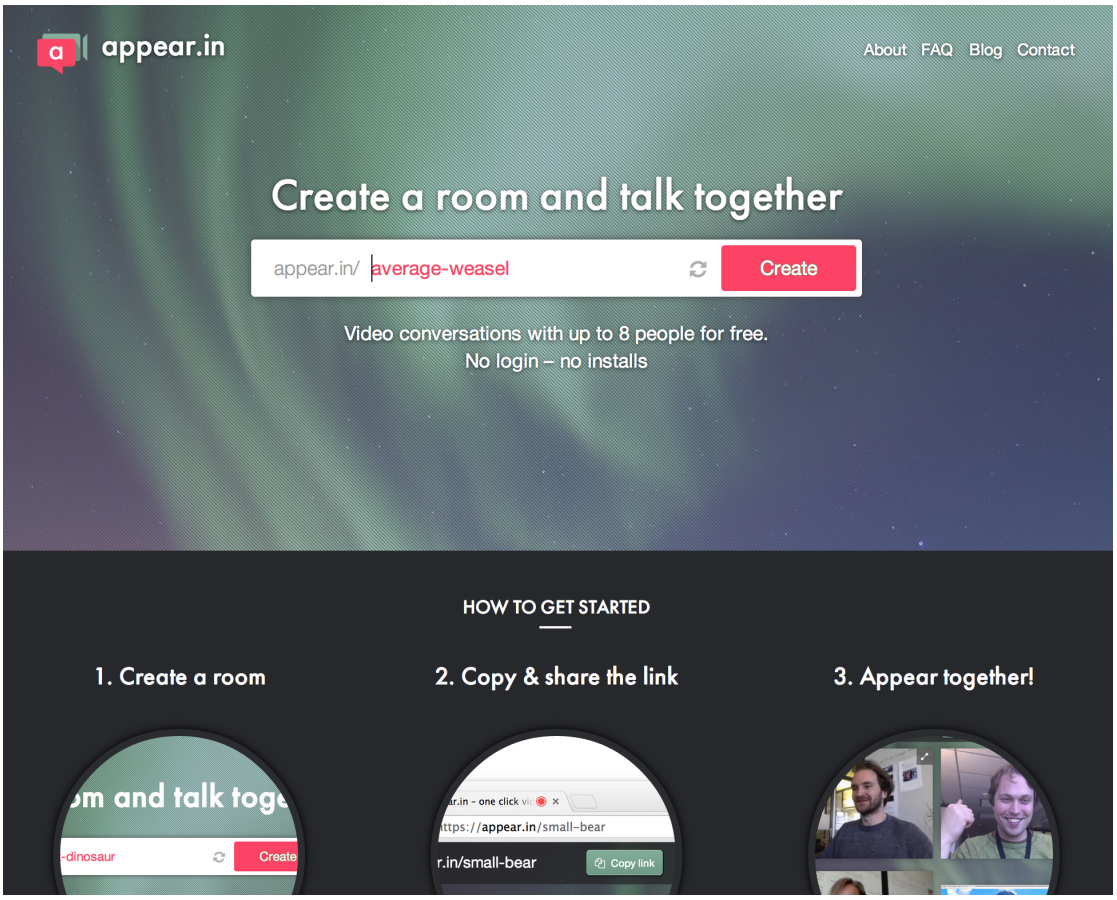


FIGURE 2.2: The appear.in landing page (as of May 2014).

The concept of evoking interest and of activating the user are universal terms that should generalize well to other web applications.

2.1.1.2 The room page

The room page, the actual product user interface, is composed of several parts. Each participant resides in his or her own "window", and various room controls are placed along the top and bottom parts of the page.

As the quality of the video conferencing part of the application is largely governed by the browser and other low-level technicalities, we will mostly focus our efforts on the functionality augmenting the content: effectively, the rest of the UI.

Please note that all features discussed in this section are continuously subject to heavy revision, and should not in any way be viewed as a permanent or final set of features.

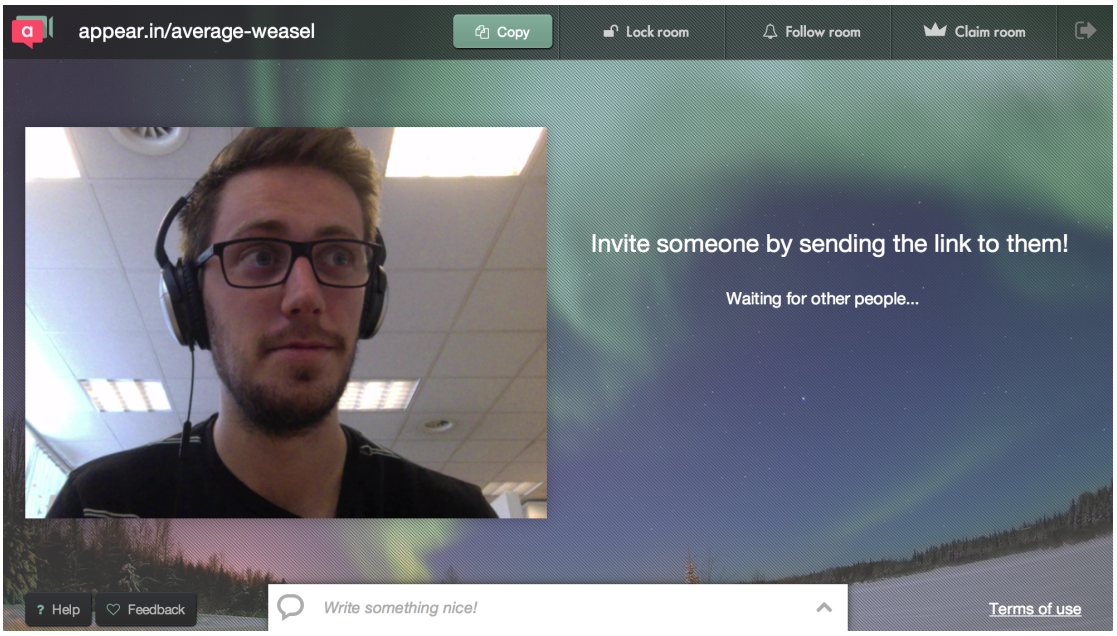


FIGURE 2.3: The author in an appear.in room (as of May 2014).

The leftmost part of the top bar consists of a URL copying control, as shown in figure 2.4a. Many users utilize this area when copying the page URL to invite their peers to the room. However, seeing as the same effect is easily achieved by copying the address field of the browser – which we cannot track – use of this control does not give a complete picture of users' sharing behavior.

To the top right is a row of buttons, as shown in figure 2.4c. Respectively, they allow the user to "lock", "follow", "claim", and leave the room. Of these, only the first three are of particular interest, as the "leave room" button essentially does nothing but close the window – which we, again, cannot track.

All these buttons alter the state of the room. Let's briefly walk through them.

Lock

When locking a room, one prevents other people stumbling upon the room's URL from entering. They are, however, able to knock their way in, much in the same fashion as one would enter a locked room in the physical world.

Follow

Users following a room are notified when other people enter it.

Claim

Users can claim a previously unclaimed room, and essentially take ownership of it. This enables them to customize the room in a number of ways.

The last piece of the feature puzzle is the chat control, depicted in figure 2.4b. When users post a message, it becomes visible to other members. Chat messages are written to a centralized store, and persist as long as there are people in the room.

2.1.2 Generality of the application case

To summarize section 2.1.1, we will focus on the application features augmenting the main video conferencing functionality. These functions essentially either *manipulate the room resource* or aid in *exporting information* about it. Thus, the analyzed feature set should apply equally well to other applications where these are the central user tasks.

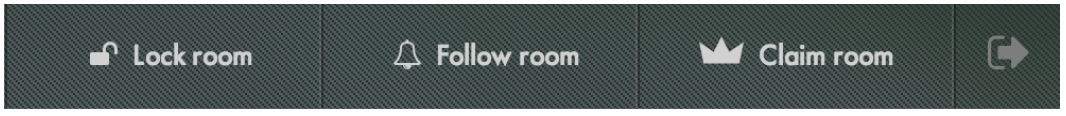
However, as touched upon in section **??**, the main challenge with appear.in as an adaptation use-case is 1) the absence of demographics, and 2) the unreliability of user identity.

2.1.2.1 The absence of demographics

@TODO: refer to work involving demographics (surveys, webshops, marketing foobar etc.), work not requiring it (clickstream analysis, implicit user modeling etc), and work just talking about the issue.

(A) The room URL copying control.

(B) The chat control.



(C) The top button row. Each button serves its own purpose and fires its own event.

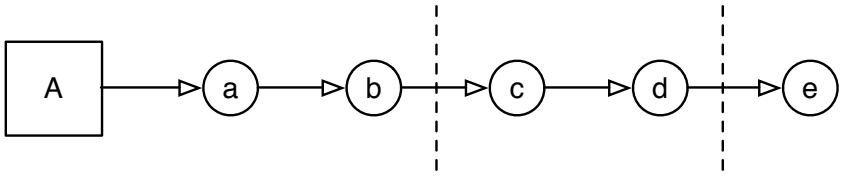
FIGURE 2.4: The most important UI parts.

The absence of demographics leaves us with less information for us to found our predictions on.

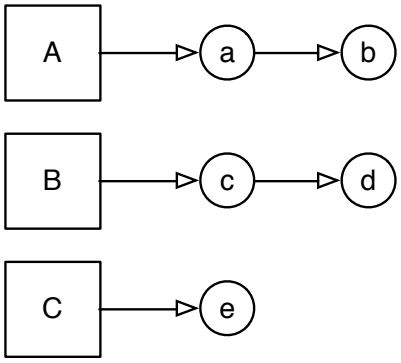
2.1.2.2 The unreliability of user identity

Being an anonymous service, appear.in has no explicit login. As explained in section 1.2.4.2, appear.in uses cookies to track users over time, which introduces some challenges when it comes to building user models.

Users periodically tend to clear browser cookies, which causes an abrupt end of the perceived event stream from the browser user, never to be reassumed. The effect is illustrated in figure 2.5 for a hypothetical case, in which user A clears the browser cache twice, effectively cutting off the event stream each time. There is no obvious way in which to consolidate the three event streams of the three perceived users A, B, and C.



(A) An event flow from $a \rightarrow e$, as seen from a user A. The dashed vertical lines indicate a point at which the user clears the browser cache.



(B) The same event flow as perceived by the system.

FIGURE 2.5: The impact that clearing the browser cookies has on the chronological event stream for a single user A.

In practice, this leads to sparse usage data and quite a bit of noise in the event stream used as the foundation for the user model generation.

However, the shortening of event streams also introduces a bias to the clustering algorithms, as the user model vectors will be shorter than is actually the case. We will return to this issue in section 3.4.

These are all issues that apply especially to appear.in, but that may also apply to many other anonymous systems.

2.2 The field of user adaptation

This section will present an overview of the field of user adaptation. From the rather short history of the field, we move on to the conceptual frameworks and methodologies that modern adaptive systems base themselves on, before surveying the state of the art applications.

2.2.1 A brief history of adaptive systems

Vrieze discerns the history of adaptive systems into three eras [6]: early research, the pre-Internet era, and the Internet era. His historical dissertation is loosely based on Kobsa [7]. This summary will do the same, but focus on what Vrieze calls *the Internet era*.

2.2.1.1 Early research

The first work within user modeling research was conducted in the nineteen-eighties. Strongly influenced by the field of artificial intelligence, the groundwork for modern user modeling was laid.

The common denominator for the user modeling systems composed in this era was their tight integration with their respective production systems. The end of the era, however, saw systems such as GUMS [8], the first standalone user modeling system. These early standalone systems are most widely known as User Modeling Shell Systems, a term coined by Kobsa in 1990 [9].

2.2.1.2 Pre-Internet research

The nineteen-nineties was a decade widely dominated by User Modeling Shell Systems. Mostly, stereotype-based approaches were used, which sought to deduce logical connections between user models and the application domain.

One system, however, Doppelgänger [10], stands out in this regard, being the only system to employ a probabilistic model, and not a logic-based approach [7**? ?**].

2.2.1.3 Internet-time research

@TODO

–

What is it? Other domains where it's used (marketing).

How'd it start? What use cases have been popular throughout its history? What are recent developments and key enablers?

2.2.2 Personalization on the web

When the field of user adaptation meets the web, we have traditionally tended to dub it either "web personalization" or "adaptive hypermedia". The terms are defined as follows:

Web personalization

> Any action that adapts the information or services provided by a Web site to the needs of a particular user or a set of users, taking advantage of the knowledge gained from the users' navigational behavior and individual interests, in combination with the content and the structure of the Web site [11].

Adaptive hypermedia

> A hypertext or hypermedia system, with a user model, able to adapt the hypermedia using the model [12].

The first definition assumes the website to be a hierarchy of content, whereas the second definition strongly relates to the concept of hypertext and hypermedia. However, large parts of the modern web have indeed shifted towards the application domain, as discussed in section 1.1.1, where the traditional concept of a website being some kind of structured set of "content pages" is completely outdated. In this light, these definitions are both a bit dated.

Recent research, however, usually does not make these kinds of assumptions about the form or structure of the systems being adapted, and is mostly quite applicable for the case of modern web applications.

Thus, we will be using the terms "user adaptation" and "personalization" without relating it to the domain in which it is visible to the user, as this is quite irrelevant with a clear separation of the application and adaptational component. The evolution of this model through the 90s and into the modern internet era is elaborated on in the following sections.

2.2.3 Conceptual framework

2.2.4 Implementation methodology

As mentioned, there are many approaches to the task of user adaptation. Due to the constraints outlined in the previous chapters, we will be honing in on the approach termed "web usage mining", or sometimes "clickstream analysis".

Traditionally, this has entailed tracking the way users traverse a website hierarchy, looking for path patterns among them, and using this information to adapt the pages in various ways [11, 13, 14]. While not entirely the same, this is mostly analogous to the type of adaptation problem we are solving, and the systems proposed in this earlier work solves many of the same problems that we will need to solve.

A general, high-level sketch of a typical adaptive system is outlined in figure 2.6.

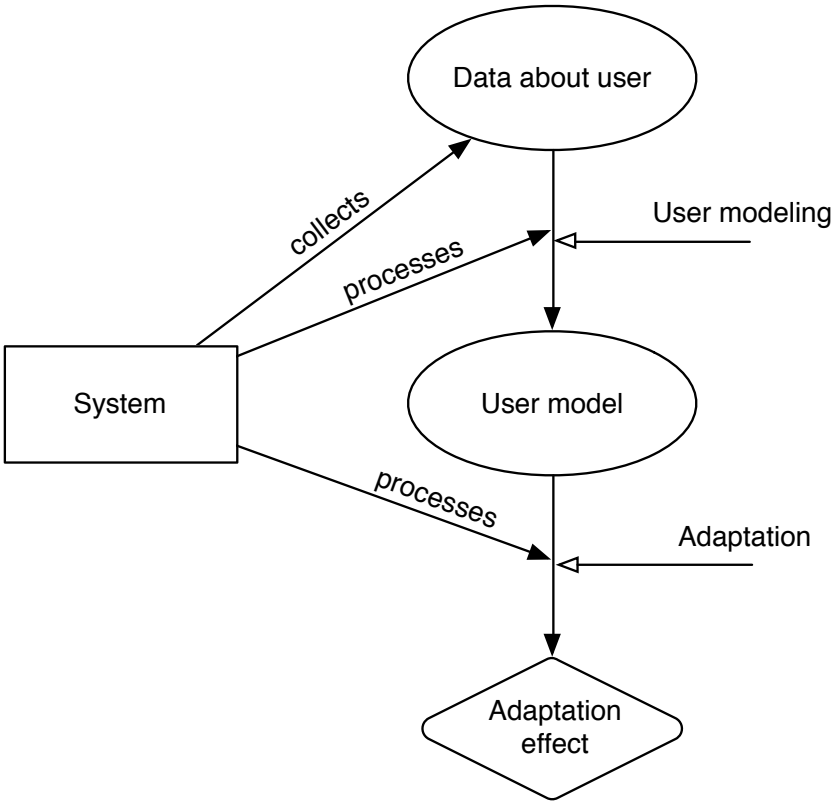


FIGURE 2.6: Caption.

2.2.5 Privacy versus personalization

2.3 Segmentation techniques

There are several ways of

2.4 A/B testing

2.5 State of the art

2.6 Similar research

The approach taken to adaptive personalization is based heavily on the work by Vrieze in "Fundaments of Adaptive Personalization" [6].

2.6.1 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 [2, 3]. 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 3.4.

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.

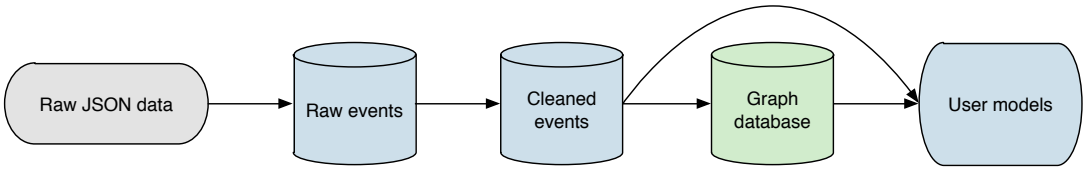


FIGURE 3.1: The ingestion pipeline broken into 4 steps. The color of each node indicates means of storage: *Blue* indicates a RDBMS, *green* indicates a graph database, whereas *gray* is used to indicate flat file storage.

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 1.2.4, 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.

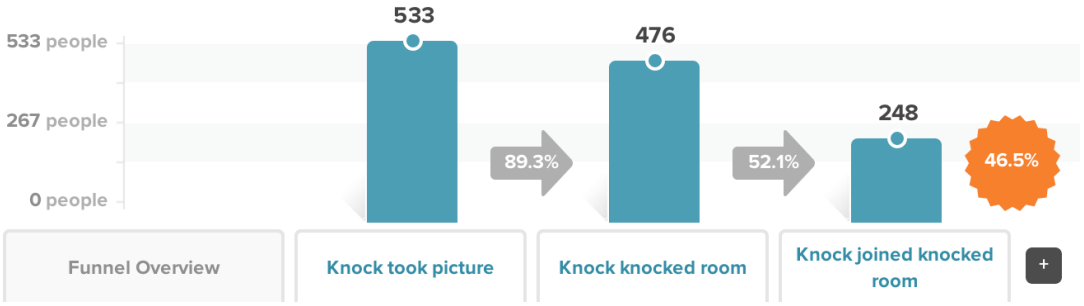


FIGURE 3.2: Example of a simple funnel report.

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 [15] 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 [16].

@TODO: Remember the bias introduced in section 2.1.2.

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.

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.

```
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

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 3.5, 3.4, and 3.6:

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.

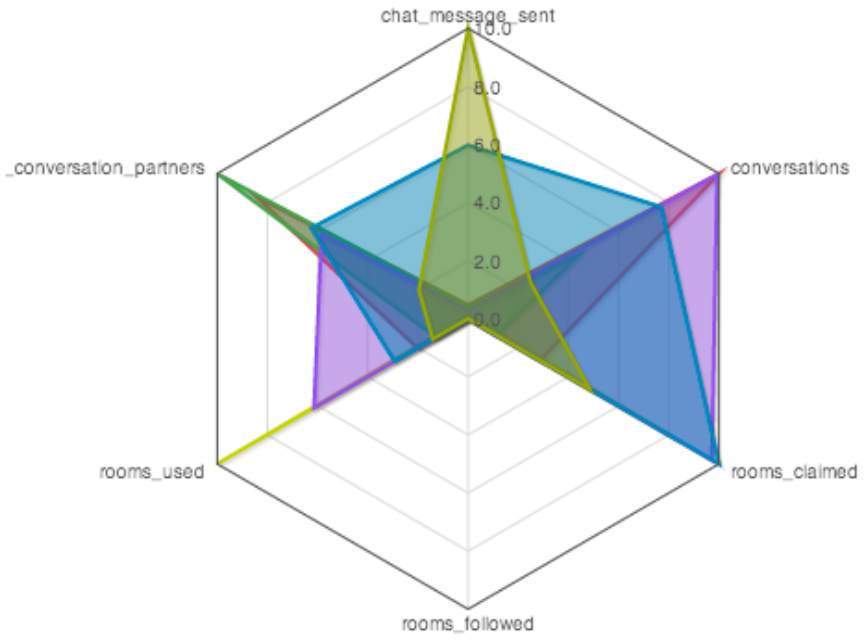


FIGURE 4.1: Radar chart comparing clusters generated from data for February 2014.

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

@TODO: Should this part be moved to appendix?

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

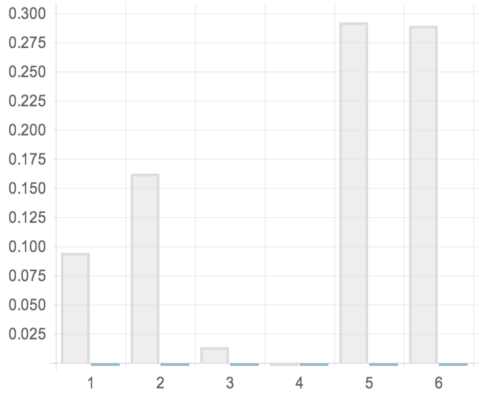
As shown in figure 4.2b, the users in this cluster score close to 0 on every feature except the number of rooms used – most notably, the number of conversations.

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

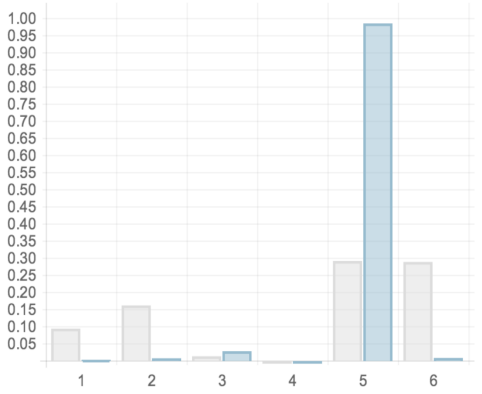
4.4.1.3 Cluster 3: Simple users with small networks

Users in cluster 3 (see figure 4.2c) 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.

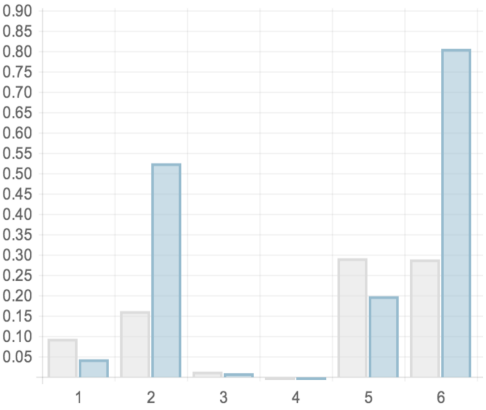
Persona 3. A returning user, only utilizing the bare functionality, conversing only with a very limited group of people.



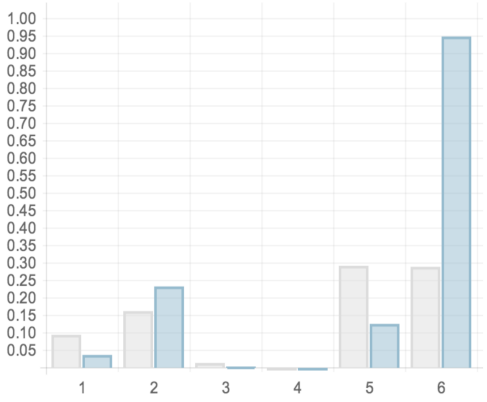
(A) Cluster 1: Front page hitting users



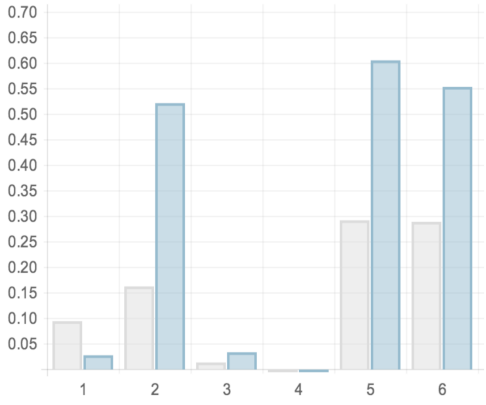
(B) Cluster 2: Users trying out the service alone



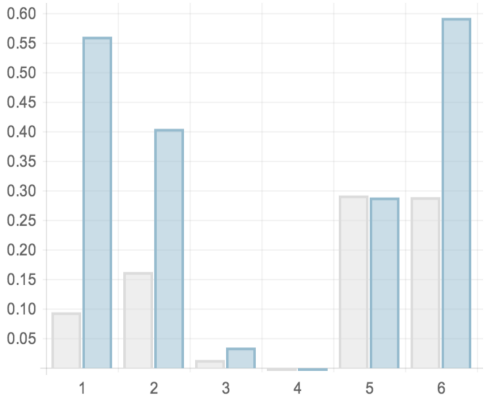
(C) Cluster 3: Simple users with small networks



(D) Cluster 4: Simple users with large networks



(E) Cluster 5: Incognito users



(F) Cluster 6: Chatty, simple users

FIGURE 4.2: Cluster centers compared to the unweighted average cluster centers. The features plotted are 1) chat messages sent, 2) conversations, 3) rooms claimed, 4) rooms followed, 5) unique rooms used, and 6) conversation network size.

4.4.1.4 Cluster 4: Simple users with large networks

The users in cluster 4 (see figure 4.2d) are very much like the ones in cluster 3, but use the service slightly more, and are part of much larger networks.

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

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 4.2e.

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

4.4.1.6 Cluster 6: Chatty simple users

The users of cluster 6 are very much like those in cluster 3, as can be seen in figure 4.2f. The significant difference is that they make heavy use of the chat functionality.

Persona 6. A user sharing the characteristics of users in cluster 3, except in making use of the text chat functionality.

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.

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 [17].

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] W3C. Html5, a vocabulary and associated apis for html and xhtml, introduction, 2014. URL http://www.w3.org/TR/html5/introduction.html.

[2] Maximilian Teltzrow and Alfred Kobsa. Impacts of user privacy preferences on personalized systems. *. . . personalized user experiences in eCommerce*, 5(0308277), 2004. URL http://link.springer.com/chapter/10.1007/1-4020-2148-8_17.

[3] Alfred Kobsa. Privacy-Enhanced Web Personalization. *Communications of the ACM*, 50:628–670, 2007.

[4] 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.

[5] Edsger W Dijkstra. *Selected writings on computing: a personal perspective*. Springer-Verlag New York, Inc., 1982.

[6] PT De Vrieze. *Fundaments of adaptive personalisation*. Paul de Vrieze, 2006. ISBN 9789090211138. URL http://books.google.com/books?hl=en&lr=&id=9wyclzI91McC&oi=fnd&pg=PA1&dq=Fundaments+of+Adaptive+Personalisation&ots=qDOGKmb6hw&sig=b51fHFbRmGXV4nJ1EjkH3-XrvHI.

[7] Alfred Kobsa. Generic User Modeling Systems. pages 49–63, 2001.

[8] TimothyW. Finin. Gums — a general user modeling shell. In Alfred Kobsa and Wolfgang Wahlster, editors, *User Models in Dialog Systems*, Symbolic Computation, pages 411–430. Springer Berlin Heidelberg, 1989. ISBN 978-3-642-83232-1. doi: 10.1007/978-3-642-83230-7_15. URL http://dx.doi.org/10.1007/978-3-642-83230-7_15.

[9] Alfred Kobsa. Modeling the user's conceptual knowledge in bgp-ms, a user modeling shell system1. *Computational Intelligence*, 6(4):193–208, 1990. ISSN 1467-8640. doi: 10.1111/j.1467-8640.1990.tb00295.x. URL http://dx.doi.org/10.1111/j.1467-8640.1990.tb00295.x.

[10] Jon Orwant. Heterogeneous learning in the Doppelgänger user modeling system. *User Modeling and User-Adapted Interaction*, 4(2):107–130, 1995. ISSN 0924-1868. doi: 10.1007/BF01099429. URL http://link.springer.com/10.1007/BF01099429.

[11] Magdalini Eirinaki and Michalis Vazirgiannis. Web mining for web personalization. *ACM Transactions on Internet Technology*, 3(1):1–27, February 2003. ISSN 15335399. doi: 10.1145/643477.643478. URL http://portal.acm.org/citation.cfm?doid=643477.643478.

[12] Peter Brusilovsky. Methods and techniques of adaptive hypermedia. *User Modeling and User-Adapted Interaction*, 6(2-3):87–129, 1996. ISSN 0924-1868. doi: 10.1007/BF00143964. URL http://dx.doi.org/10.1007/BF00143964.

[13] Bamshad Mobasher, Robert Cooley, and Jaideep Srivastava. Automatic Personalization Based on Web Usage Mining. 43(8), 2000.

[14] Alan L. Montgomery and Michael D. Smith. Prospects for Personalization on the Internet. *Journal of Interactive Marketing*, 23(2):130–137, May 2009. ISSN 10949968. doi: 10.1016/j.intmar.2009.02.001. URL http://linkinghub.elsevier.com/retrieve/pii/S1094996809000322.

[15] 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.

[16] Michael J Berry and Gordon S Linoff. *Data mining techniques: for marketing, sales, and customer relationship management*. John Wiley & Sons, 1997.

[17] 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.