Norwegian University of Science and Technology

Master's thesis, spring 2014

User adaptation in anonymous web applications

Author:
Jonas Myrlund

Supervisors:
Prof. Heri Ramampiaro
Prof. Helge Langseth

June 2014

"C is for cookie, and cookie is for me; C is for cookie, and cookie is for me."

Cookie Monster, Sesame Street

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
anonymous web applications**

by Jonas Myrlund

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

In this project I 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 how different classes of users use an application differently 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.

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 uncertain to what extent the results generalize.

Acknowledgements

First and foremost, I would like to thank my supervisors, Heri Ramampiaro and Helge Langseth. While providing me with plenty of space to do things my own way, they have been an immense help in finding the academic angle for the project, and in continuously steering me back on course every time I've lost track of the goal.

Telenor Digital AS have allowed me to write this thesis with them, and for this I am very grateful. Especially big thanks to Svein Yngvar Willassen, my supervisor at Telenor Digital, who both talked me into doing a master's project with appear.in, and assisted me greatly along the way.

Obviously, a big thanks to the entire appear.in team. It is extremely inspiring to do research when you thoroughly believe in the application you are working to improve. Ahead of me throughout the entire project period, they have improved the application immensely and attracted huge amounts of users who generated lots of data – without which this thesis would have been impossible to write!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and motivation

The advent of HTML5 enables us to build increasingly sophisticated applications that run right in the web browser on demand.

Apart from the tendency to move ever closer to feature parity with native applications, most web applications share a common trait seldom seen elsewhere: they are available instantly and on-demand. Although some applications require the user to sign up or in other ways get past a paywall, the notion of a web application as being available *without installation* remains. This is discussed further in 2.1.1.

This lack of friction is something many web applications leverage. Indeed, the main competition among applications is often a matter of minimizing friction: a push towards simplicity and ease-of-use. This often involves the absence of authentication. As the users and the legislative forces governing the Internet are becoming more and more privacy-aware, there is little reason to believe this application niche is going away in the near future (see 2.2.4 for discussion).

A real-world application adhering to the concept of minimizing friction is the application case for this project; appear.in[1] is an attempt at providing a full-fledged video conferencing service with as little friction as possible. Among other things, this entails the application having no particular notion of users, and with that a highly unstable notion of *identity*.

An interesting question is, then, what can we do with the unstable data situation at hand? Is the behavioral data we have available enough to be able to do any significant

[1] Available at `https://appear.in`.

user adaptations, for instance? In very broad terms, this is what this project sets out to answer.

1.2 Research questions

This project will investigate whether users of a simple service, like the appplication case, fall into clear sterotypical patterns. Further, it will attempt to measure to what extent these user stereotypes can be used as a basis for user adaptations.

Since these problems in many ways build on each other, they will have to be answered in sequence. More specifically, I will attempt to answer the following questions:

1. Is it possible to consistently stereotype the users of simple web applications without requiring explicit authentication?
2. Are these stereotypes usable as a basis for user adaptations within the application?

The problem of stereotyping users will involve generating user models. However, as the application does not have a clear notion of a user, much of the discussion will be dedicated to dealing with the ways we can and cannot circumvent this problem.

1.3 Organization of the thesis

This paper is organized as follows.

Chapter 2 surveys the application case and the available data, discusses some important identification issues, and surveys relevant research. Chapter 3 describes the approach taken to answer the research questions. In chapter 4 an implementation of the approach described in chapter 3 is evaluated, before chapter 5 discusses some important takeaways, and suggests further work.

Chapter 2

Survey

This chapter will present the application case and the available data, discusse some important identification issues, and survey relevant research within the areas to be touched upon in the approach taken to answer the research questions.

2.1 The application case

This section will introduce the application case, appear.in. First, to understand why it is of particular interest to study it, we need some context.

2.1.1 The modern web

When we say that web applications are nearing the power and feature parity of traditional desktop applications, we are of course talking about the introduction of HTML5.

HTML was originally designed as a language for describing scientific documents, and little more. Through the last decade, however, the concept of web applications slowly established itself, while lacking clear standardization efforts from the W3C. The HTML5 specification is an attempt to remedy this, by providing standards and guidelines for the browser vendors on how to implement a wide range of common APIs [1].

In practice, these APIs lets websites do things like:

- play audio and video[1]
- generate graphics[2]

[1] http://dev.w3.org/html5/spec-author-view/video.html
[2] http://www.w3.org/TR/2dcontext/

- access your webcam and microphone[3]
- handle and manipulate arbitrary files[4]
- send and recieve data over full-duplex socket connections[5]

...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.

2.1.2 The WebRTC specification

One of the major HTML5 API specifications is called WebRTC[6], and it is the last piece of the API puzzle enabling applications like appear.in.

As the name implies, WebRTC handles real-time communication, but for the case of appear.in, an important aspect is that it is able to do so peer-to-peer. Although designed to be a protocol for exchanging arbitrary data between peers, it is particularly geared toward multimedia. 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[7].

Although the WebRTC specification at the time of writing still officially is a working draft in the W3C[8], 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.

2.1.3 Introducing appear.in

The main idea behind appear.in 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. There is no notion of *calling* someone – you simply meet up in a room and talk. 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.

[3] `http://www.w3.org/TR/mediacapture-streams/`
[4] `http://www.w3.org/TR/FileAPI/`
[5] `http://www.w3.org/TR/websockets/`
[6] Web Real-Time Communication.
[7] For an excellent introduction, see: `http://www.html5rocks.com/en/tutorials/webrtc/basics/`.
[8] The latest specification can be found here: `http://dev.w3.org/2011/webrtc/editor/webrtc.html`.

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, the "room", to be identifiable by a string of characters.

Until appear.in, this particular way of thinking about audiovisual conversations hasn't been a commonly seen pattern. However, the simplicity of the room concept opens the service up for a wide variety of uses: 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.

2.1.4 The inner workings of appear.in

This section will pick apart the workings of appear.in to allow for a thorough understanding of what kind of an application we are dealing with. This should provide a good basis for understanding the generated data and the applicability of the results for the general case.

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 service endpoint.

The instrumentation service has been included simply to illustrate the fact that the available data is generated and logged *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.
- 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.

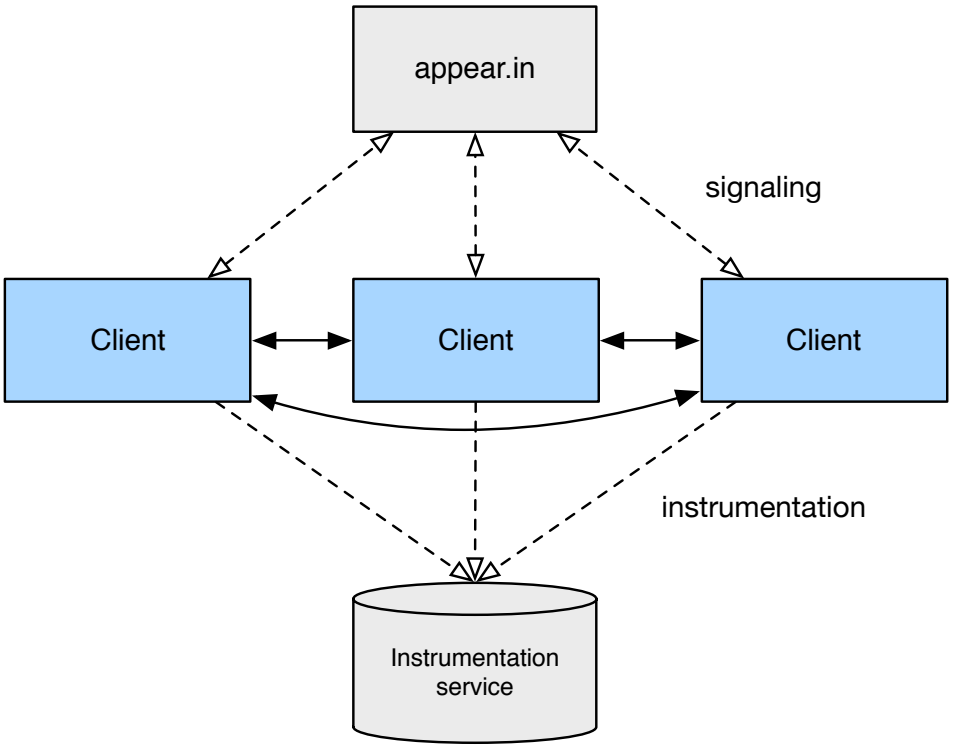


FIGURE 2.1: The appear.in architecture, illustrated for a conversation between 3 peers. The black arrows indicate media data flow, and the dashed arrows indicate metadata flow: signaling data and instrumentation data, respectively.

2.1.4.1 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).

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

2.1.4.2 The room page

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

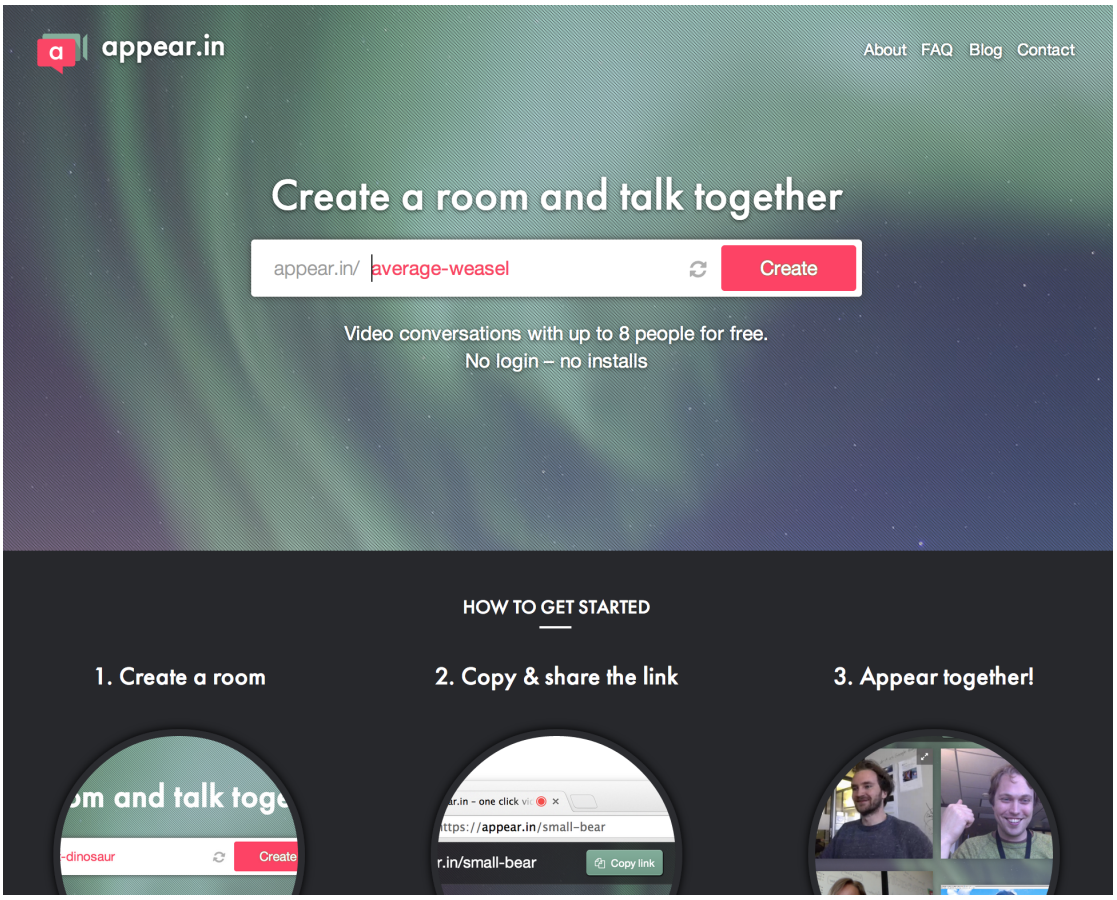


FIGURE 2.2: The appear.in landing page (as of June 2014).

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.

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 2.4b. 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, severing the connection.

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

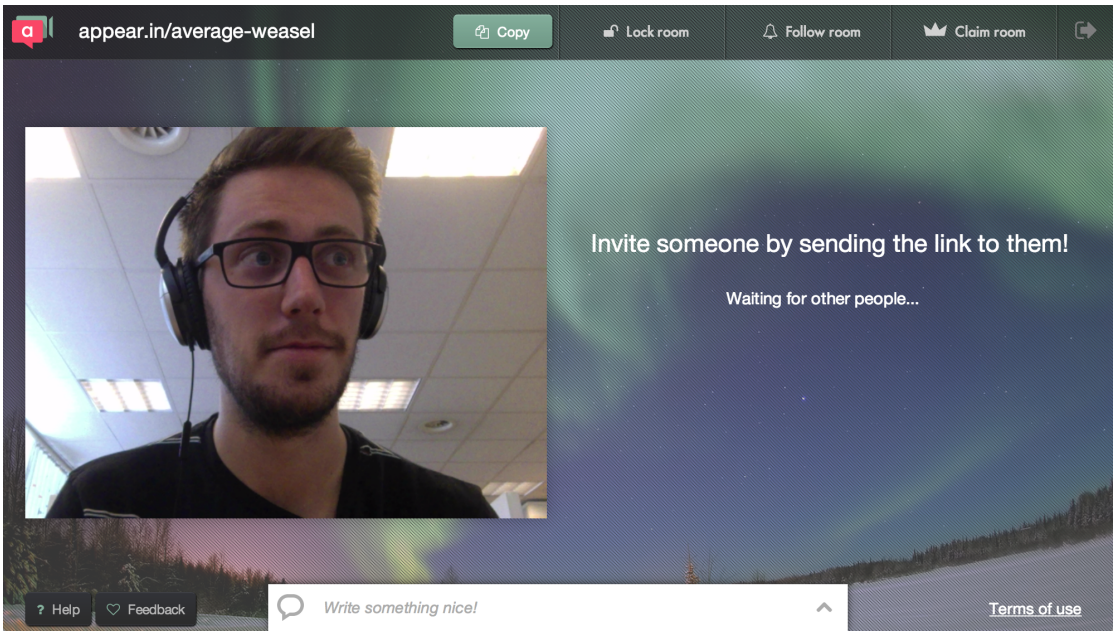


FIGURE 2.3: The author in an appear.in room (as of June 2014).

Lock

When locking a room, one prevents other people who stumble upon the room's URL from entering.[9]

Follow

Users following a room are notified when other people enter it. A room's followers may also follow the room chat without being present in the room, by using a browser extension.

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.

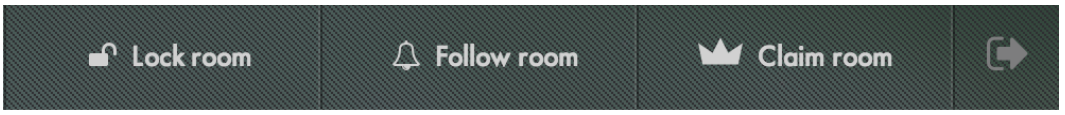
The last piece of the feature puzzle is the chat control, depicted in figure 2.4c. When users post a message, it becomes visible to other members. Chat messages are written to a centralized store, and persist as long as there are people in the room.

2.2 The quality of the data

@TODO

[9] 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.

(A) The room URL copying control.



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

(C) The chat control.

FIGURE 2.4: The most important UI parts.

2.2.1 Lack of demographics

A key facet of all user adaptation and personalization is adapting to a user's interests, and so it is imperative to learn about the user. Montgomery and Srinivasan introduce a distinction between active and passive learning to aid in categorizing approaches to the issue [2]. Whereas active learning results from direct questions to the user, passive learning is the opposite: learning about the user without asking.

Active learning has several disadvantages in the general case:

1. It requires too much effort on the customer's part.
2. The user may indeed not know the answer to the questions, either lacking the proper knowledge or experience to evaluate the alternatives.
3. The user may be unwilling to reveal correct answers.
4. It is inefficient, as it typically ignores information consumers reveal about their preferences in their past interactions and purchases.

The first point applies especially to the case of appear.in. An important part of the product is the simplicity of the application, ie. the small amount of friction. Thus, introducing questionnaires or similar approaches to collecting active feedback from users has not been viewed as a positive tradeoff up to this point.

So, we will be limited to passive learning. What does this leave us with in practice?

Montgomery identifies three major sources of information from which it is possible to learn passively: transaction data, clickstream data, and email.

appear.in, being a so-called single-page web application (SPA), has no clickstream in the traditional sense, a traditional clickstream being the series of pages navigated to. Transactional data, on the other hand, is usually related to e-commerce, and to some sense of "items bought". This is similarly irrelevant in this particular interpretation. However, we *can* choose to view the series of interactions with the application as a clickstream of sorts – an event stream – and use it as a data source in much the same way.

2.2.2 What is a user identity?

Before moving on, let's define some words and concepts that will be central to this section, and to the rest of the thesis. The following definitions have been taken from the Merriam-Webster online dictionary[10].

Anonymous
: Lacking individuality, distinction, or recognizability.

Identity
: The distinguishing character or personality of an individual.

Pseudonymous
: Using a pseudonym: a name that someone uses instead of his or her real name.

Adapted to our realm, this thesis will use these words in the following ways.

Anonymous
: Not being recognizable as a person from the collected user data.

Identity
: The ability to be distinguished from other users over time.

Pseudonymity
: Being able to identify users without actual personal information.

In these word senses, 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.

[10] http://www.merriam-webster.com/dictionary/

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.

2.2.3 Client-side identity

Several others have written about how privacy constraints impact personalized systems [3, 4]. Indeed, 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 [5]. 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.

This all makes tracking users very problematic. However, we can somewhat circumvent the problem using web cookies, albeit with a very time-limited effect.

2.2.3.1 Preserving state with cookies

The first time a user visits appear.in, a cookie is set with a *random value* uniquely identifying the *browser*. It is this value which is sent with every event to the instrumentation service, as described in figure 2.1.

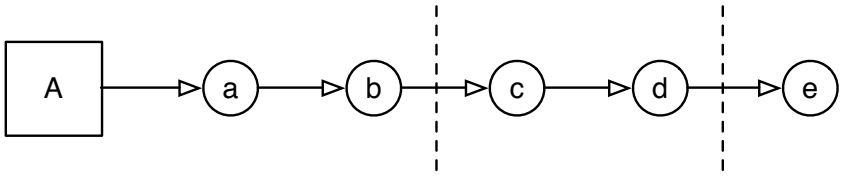
Unfortunately, using only cookies for identification has its clear downsides. Should the user clear the browser cache, switch browsers, use the browser "incognito"[11], or simply use multiple machines, then different user ids will be generated for each case.

Without tracking more user information – IPs and locations, for instance – there is no easy way of tying these user ids together, 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.

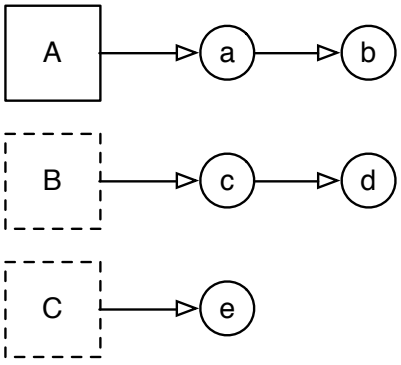
As we shall see, most users periodically clear their browser cookies (see 2.2.4 for numbers), which causes an abrupt end of the perceived event stream from the browser user, never to be reassumed. The effect is illustrated in figure 2.5 for a hypothetical case, in which user A clears the browser cache twice, effectively cutting off the event stream

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

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 basis for the user model generation. Furthermore, 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.

2.2.4 Privacy versus personalization

There are not only the technical sides of user tracking to deal with. Due to various non-technical reasons, there exists at the time of writing a great deal of controversy surrounding the issue of online privacy.

This section discusses how this affects our current ability to perform effective user adaptations, and some prospects for the future.

Teltzrow and Kobsa [3] state the issue plainly: "Personalization systems need to acquire a certain amount of data about users' interests, behavior, demographics and actions before they can start adapting to them." As we shall see, many Internet users are highly sceptical of providing personal information to web sites, and a majority are concerned

about web sites tracking their movements and behavior online. This doesn't fit adaptive systems' demand for data collection.

2.2.4.1 Personal information

First, consider the following survey results regarding personal infomation [3]:

1. Internet Users who are concerned about the security of personal information: 83% [6], 70% [7], 84% [8]
2. People who have refused to give (personal) information to a web site: 82% [9]
3. Internet users who would never provide personal information to a web site: 27% [8]
4. Internet users who supplied false or fictitious information to a web site when asked to register: 34% [9], 24% [8]

Although the above numbers aren't directly relevant to the case of appear.in, where no personal information is collected or stored, it underlines a general scepticism towards providing information to web sites.

The fact that such a large portion of users are sceptical of providing personal data tells us that there is reason to believe there is room for anonymous niches within most application areas, serving as a drive towards more applications like appear.in.

2.2.4.2 Tracking

While there is no theoretical upper bound to the lifetime of a tracking cookie, surveys of Internet users, as well as our own analyses, see that they usually do not persist for very long [3].

1. People who are concerned about being tracked on the Internet: 60% [6], 54% [8], 63% [10]
2. Internet users who generally accept cookies: 62% [11]
3. Internet users who set their computers to reject cookies: 25% [9], 3% [6], 31% in warning modus [6], 10%[8]
4. Internet users who delete cookies periodically: 52% [11]

These numbers are backed up by looking at the decline over time of user cookies in appear.in, as shown in figure 4.7.

2.2.4.3 The road ahead

Although these numbers are a bit dated, there is little reason to believe that the tracking situation is going to get any easier in the years to come [12–15].

As we have seen, there already exists broad scepticism towards the use of cookies to track users, even on first-party sites. Much of the problem seems to stem from sites allowing third-party cookies, to better serve advertisements to its users – who most often are oblivious to the tracking taking place at all.

These third-party cookies, however, allow these third-parties to track users' activity and behavior across multiple sites, often without their explicit consent. This has recently been deemed as bordering to surveillance, and in recent years extensive legislative restrictions have been introduced to decrease the prevalence of particularly third-party cookies.

This all presents a challenge for services like appear.in, who use third-party cookies not to advertise, but to enhance the service for the user. There is reason to believe this situation will not become easier to deal with in the years to come, but hopefully, there will come about better ways of dealing with the issue.

2.3 Identifying stereotypes

There are many approaches to the task of sterotyping users. In this project we will approach the task using standard clustering techniques.

In the broad sense of the problem at hand – adapting the application – there are, of course, alternatives to the clustering approach taken. Two of them are *classification* and *collaborative filtering*:

Collaborative filtering

Let us briefly reiterate on the first main research question: is it possible to consistently stereotype the users? While collaborative filtering may be a good approach to the user adaptation problem isolated, it will not be able to aid us in identifying stereotypes.

Classification

Classification problems require training sets with predefined stereotypes. As there is no preexisting set of stereotypes, and no known patterns within the data, a classification approach is unfeasible.

Due to these constraints, the clustering approach is chosen. The following sections take a closer look at how we can use clustering techniques to identify and evaluate the quality of user stereotypes.

2.3.1 Stereotyping with clustering techniques

There are many distinctions that can be made between the various clustering models. The following list contains a brief introduction of some of the commonly used models.

Centroid models
: Each cluster is represented by a single vector, indicating its center.

Connectivity models
: Typically hierarchical clustering, where models are built based on distance connectivity.

Density models
: Defines clusters as dense areas in the data space.

The most common model, by far, is the centroid model. This is the clustering scheme to be used in the approach described in chapter 3.

The k-means algorithm has several strong suites with regard to our application [16]:

1. It is a good match with regard to both input and desired output for our data.
2. The implementation used in this project, the Lloyd-Forgy algorithm [17], parallelizes and scales very well.
3. Being the most commercially used clustering scheme, the results will more easily compare to those of other applications.

The algorithm itself and its generated results are described in detail in 3.4.

2.3.2 Evaluating clusters

In general, any clustering method should search for clusters whose members are close to each other and well separated. Berry and Linoff [18] formulate it in terms of *compactness* and *separation*.

Compactness

The members of each cluster should be as close to each other as possible.

Separation

The clusters themselves should be widely spaced.

For an algorithm such as k-means, which takes the k as an input, the central question when it comes to cluster validity is for which value of k the cluster compactness and separation are optimal.

The most efficient way of measuring cluster quality between several executions of a clustering method, where only its parameters – like k – differ, is to utilize a so-called *relative criteria* [19].

One of the most widely used measures of clusters' relative criteria is the Davies-Bouldin index, which is the one used to differentiate between clusters in this project. It is defined as:

$$\mathrm{DB}_k = \frac{1}{k} \sum_{i=1}^{k} \max_{i \neq j} \left(\frac{s_i + s_j}{d(c_i, c_j)} \right) \tag{2.1}$$

where k is the number of clusters, c_x is the centroid of cluster x, s_x is the average distance of all elements in cluster x to centroid c_x, and $d(c_i, c_j)$ is the distance between centroids c_i and c_j.

In plainer words, the Davies-Bouldin index formulates a measure of the quality of cluster separation and compactness, while considering the number of clusters in a responsible manner. Thus, it is well suited to distinguish between runs of eg. the k-means algorithm, where the value of k differs from run to run.

2.4 Adaptive systems

Adaptive systems have been around for quite some time, and they exist in many forms. Here, this section will focus mainly on the subject of user modeling and adaptive graphical user interfaces.

2.4.1 History of user modeling

This historical dissertation is loosely based on Vrieze and Kobsa [20, 21].

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 [22], 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 [23].

The nineteen-nineties was a decade widely dominated by User Modeling Shell Systems, which carried on the tradition of GUMS from 1989. Mostly, stereotype-based approaches were used, which sought to deduce logical connections between user models and the application domain.

One system, however, Doppelgänger [24], stands out in this regard, being the only system to employ a probabilistic model, and not a logic-based approach [21, 25, 26].

Into the new millenium, many systems began to leverage internet capabilities and were focused on the web. A special class of these systems are commercial user modeling systems. While their capabilities in terms of inference were limited, they provided other capabilities needed for commercial use. They offered features such as linking with external data sources and a client server architecture for scalability and flexibility.

Much recent research has gone into adapting mobile devices, smart appliances and homes, and development of generic user modeling tool systems [20, 27, 28].

2.4.2 Designing adaptive graphical user interfaces

As user modeling work often manifests itself as user interfaces, naturally, there has been done extensive work on how the answers provided from a user modeling system can be applied to an actual production system.

This research area differs from the purely user modeling angle, as described in the previous section, in two important ways:

1. They tend to focus on ways of applying concrete interface adaptations, and their respective effects on actual users.
2. The conclusions are often of qualitative nature, basing themselves on user testing.

Results from this kind of research tends to measure an adaptation's success not in terms of transactions and ROI, but in terms of user satisfaction and the quality of the user experience [27, 28].

As we shall see, the system described in this thesis is indeed inspired by this approach, particularly in its use of feature experiments to power user adaptations. We will come back to this point in section 3.5.

2.4.3 State of the art

@TODO: Extend and refine.

It is not many years since adaptive *menus* entered mainstream commercial, with Microsoft's Smart Menus™ and the various start menus that have been launched since Windows XP [27]. When it comes to adaptive hypermedia, the main focus of research seems to be on eLearning [20], and on adapting to different screen sizes and mobile devices [28].

Although there has indeed been extensive research into consolidating privacy concerns and personalization [4**? ?**], no significant research seems to have targeted the realm of personalizing strongly privacy-constrained web applications.

Chapter 3

Approach

3.1 System overview

The proposed system is constructed around the data flowing through it. This chapter describes each part of the system outlined in figure 3.1.

The system begins by taking the data stepwise through an ingestion pipeline, importing, filtering, and cleaning it. The steps of the ingestion pipeline are discussed in section 3.2, before moving on to the user modeling component in section 3.3.

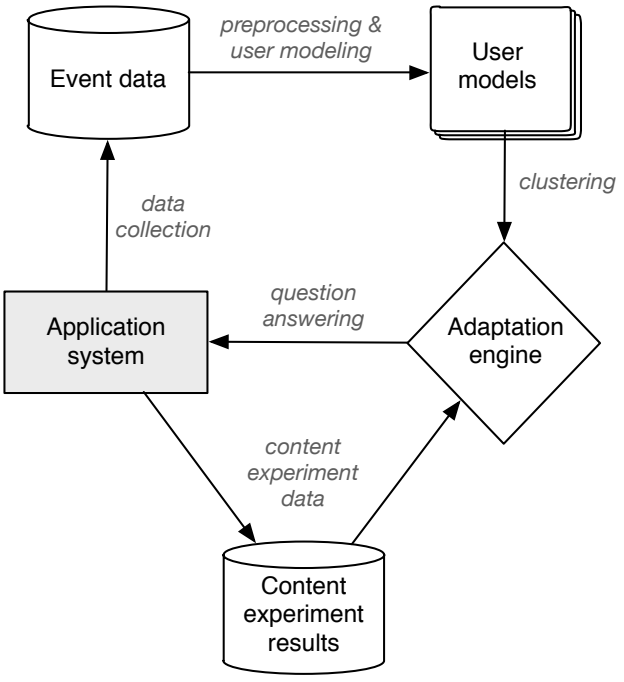


FIGURE 3.1: Overview of the system architecture.

Once user models are in place, the system identifies user segments. These are stored in a database for future use. This process is discussed in more detail in section 3.4.

Apart from clusters, the adaptation component requires content experiment results to predict how users will respond to application variations. This is discussed in section 3.5.

The components come together in the adaptation component, discussed in section 3.6, whose job it is to answer questions about users, effectively completing the feedback cycle back to the application.

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.

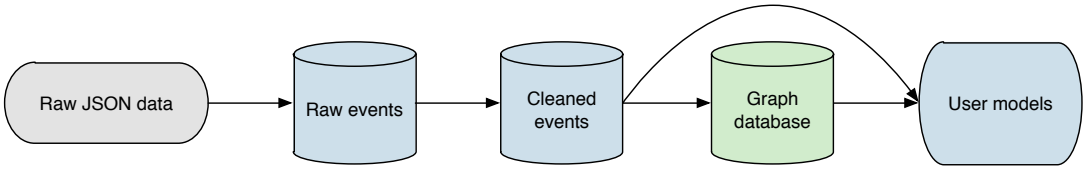


FIGURE 3.2: The ingestion pipeline broken into 4 steps. The color of each node indicates means of storage: *Blue* indicates a RDBMS, *green* indicates a graph database, whereas *gray* is used to indicate flat file storage.

3.2.1 Generating the raw data

The system input is a chronological series of raw events sent from the production system. The events are instrumented via an external analysis service called KISSmetrics[1] – a user analysis system designed around tracking individual users' behavior.

The application logs an event by calling the KISSmetrics REST API. The following data is in each instrumentation call:

1. person identifier

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

2. event name
3. user properties (optional)

The consistency of the personal identifier has already been discussed extensively in the introductory chapters, especially section 2.2.3, but a short technical introduction to the actual production system is in order.

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. 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.

Whenever something "interesting" happens, an event is sent to the KISSmetrics instrumentation service. An "interesting" event is typically 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 (example in figure 3.3) and cohort reports.

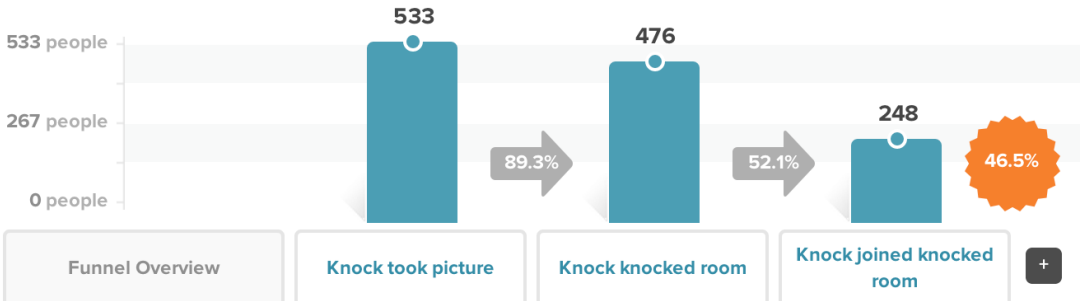


FIGURE 3.3: Example of a simple funnel report.

3.2.2 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 databases.[2]

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 numerical feature vector.

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 were mapped to their associated users for aggregation. That is, the transformation in this step started with data in the form of (3.1).

[2]To facilitate the compilation of network-related user model features, conversation data was loaded into a graph database to enable querying of network structures.

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

And ended with data in the form of (3.2), where *value* contains the aggregated value.

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

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 [29].

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 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.

Section B.1 shows a simple python-esque implementation of the k-means algorithm.

To select good parameters – the optimal value for k, given the input vectors – we will use a relative cluster validation index, like the Davies-Bouldin index discussed in section 2.3.2.

3.5 Feature experiments

Controlled experiments embody the best scientific design for establishing a causal relationship between changes and their influence on user-observable behavior [30, 31].

The simplest form of controlled experiment is often referred to as the A/B test. In A/B tests users are randomly exposed to one of two variants: control (A), or treatment (B). Based on data collected, an Overall Evaluation Criterion (OEC) is derived for each variant. Figure 3.4 illustrates the A/B testing process.

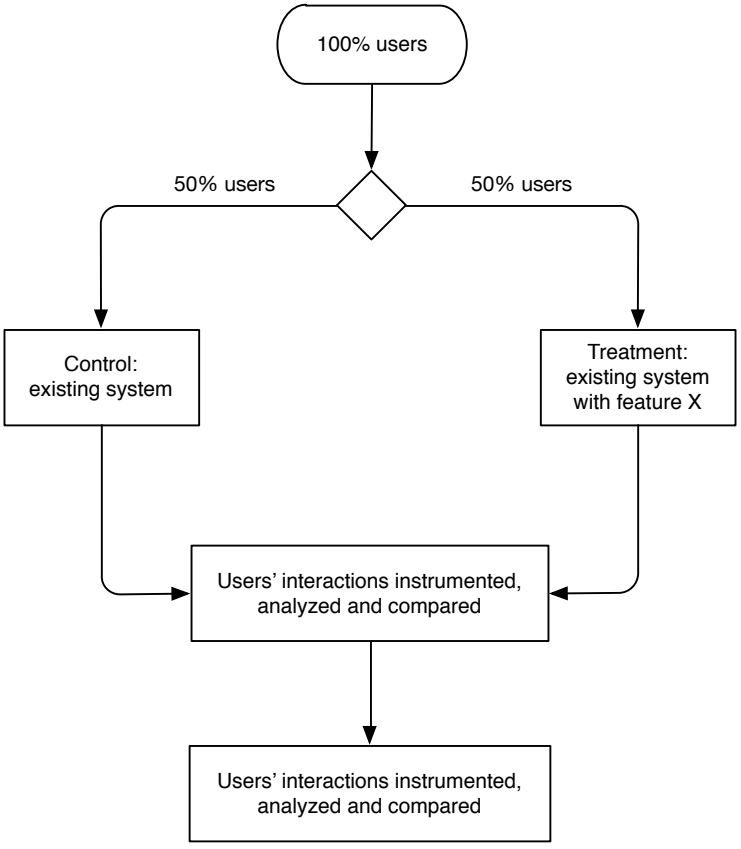


FIGURE 3.4: The flow of an A/B test.

One word in the previous paragraph, "randomly", warrants some discussion. To be able to establish a causal relationship between the selected variant and the evaluation, the variant selection must indeed be completely random, and not based on what Kohavi et al. term "any old which way" [30].

This point *must* be kept in mind when designing adaptive systems based on controlled experiments, as in our case.

3.5.1 Running a feature experiment

Ideally at this point, we have identified several significant clusters of users. Next we want to find viable ways of adapting the product to better suit each user.

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 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 proposed product alteration. An important part of this phase is to log each selected variant with the *same person identifier* as is used for other event instrumentation, as discussed in section 3.2.1.

3.5.2 Evaluation metrics

When considering which variation of a product feature to prefer, we need to be able to measure their relative success rates. As introduced above, this involves determining an overall evaluation criterion – an OEC.

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 serves as an adequate indicator of user happiness, as unhappy users have more than enough alternative applications that could 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

In the adaptation component, we want to select the variation which is most likely to provide the best experience for the user.

In other words, given a user u in cluster c and a set of variations V, we choose the variation that is more likely to maximize our selected objective function. That is,

$$\text{preferred}(c, V) = \underset{v \in V}{\text{argmax}}\, \text{score}(c, v) \tag{3.3}$$

The scoring function will in our case simply be the ratio of cluster c members who were "converted" during the experiment, ie. who achieved the designated goal, after having been presented with variation v.

$$\text{score}_{c,v} = \frac{\text{converted}_{c,v}}{\text{participants}_{c,v}} \tag{3.4}$$

Given a recent set of clusters and experiment results for the users within them, the adaptation component has what it needs to select a preferred variation based on this heuristic.

3.6.1 Applying the personalized feature set

Since well before this project, appear.in had a way of toggling features based on external criteria. The existing model allowed for overriding various settings, as well as pre-releasing features internally, using URL parameters.

We call this scheme "URL-based feature switching". Some examples are listed in table 3.1.

Parameter setting	Effect
`?video=off`	Turn off video by default.
`?audio=off`	Mute microphone by default.
`?followRoom`	Turn on experimental "follow room" functionality.

TABLE 3.1: By appending feature switches to the room URL, various effects can be achieved.

Enabling user adaptation thus became a simple matter of extending this model by allowing for feature values to be dictated by our adaptation component.

In the current codebase, the setting of each feature value happens like so:

```
var features = {
    // ...
    isVideoDisabledByDefault: $routeParams.video === "off",
```

```
    isAudioDisabledByDefault: $routeParams.audio === "off"
};
```

Here, `$routeParams` contains the parameters set in the URL, as described above. In the same way, we can perform an external call with the current user id to the adaptation component described above, and apply the desired feature set in a similar way.

Chapter 4

Evaluation

This chapter will evaluate a prototype implementation of the approach presented in chapter 3.

The chapter starts with introducing the main goal and the experimental setup in sections 4.1 and 4.2, before evaluating the actual results in sections 4.3, 4.4, and 4.5.

4.1 Prerequisites

For any of this user adaptation to have any actual use, we will need to find out whether there actually exists 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.

As described in chapter 3, a way of discovering whether this inductive bias is present in the user base is to investigate whether there is significant variance across the user groups in their users' adoption of new features. Thus, this question of whether the predictive bias exists will be included as a central part of the actual experiment itself, and be thoroughly discussed through the next sections.

4.2 Experimental setup

@TODO: Describe figure 4.1 before jumping to hypothesis.

The main hypothesis is:

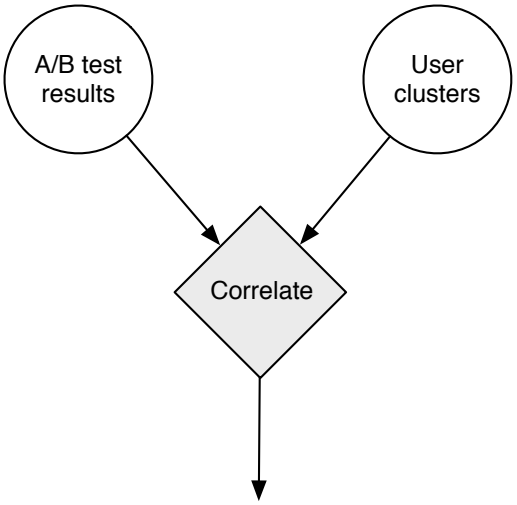


FIGURE 4.1: The experiment setup. The output of the correlation component (in gray) will determine the outcome of the experiment.

> Different variations of a service may perform differently among "different kinds of users". These variations can be used as inductive bias on which general user adaptations can be based.

Two sources of input will serve as input data to test the hypothesis:

1. A/B test results
2. User clusters

The data from the A/B test results describe each person's designated variant, and the resulting performance. See table 4.1 for some example data.

Person	Variant	Entered room	Entered conversation
p1	A	no	no
p2	A	yes	yes
p3	B	yes	no
p4	A	no	no
p5	B	yes	yes

TABLE 4.1: Each person has one assigned variant, and some performance measures.

The user cluster data contain the same persons as the test results in table 4.1 and the clusters they have been assigned to, as illustrated in table 4.2.

These data sources are aggregated with regard to variant and cluster, and the relative success rates compared.

Person	Cluster
p1	1
p2	2
p3	1
p4	1
p5	2

TABLE 4.2: Persons have been assigned to to clusters.

The next sections walk through and evaluate the actual results in the context of the experiment described in this section.

4.2.1 Evaluation case: New appear.in landing page

To test the hypothesis, we performed a controlled experiment on the user base of appear.in as we rolled out a new landing page.

The experiment stood between the old landing page, the control treatment, and a new design. We shall from here on refer to them as variation A and B, respectively. The two competing variations are depicted in figure 4.2.

4.3 A/B testing features

After running through the test period with an even randomized user split between the two variations, the results were in. In total, just over 20,000 people were included in the experiment and were served one of the two variations.

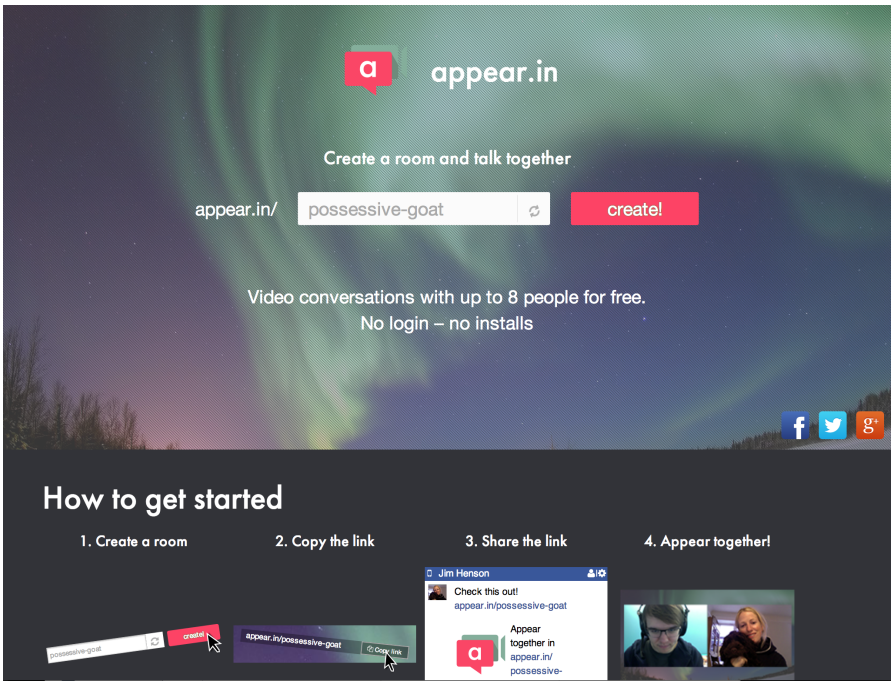
The two were compared head to head for two different performance metrics:

Visited room
: The number of users going from the landing page and into a room.

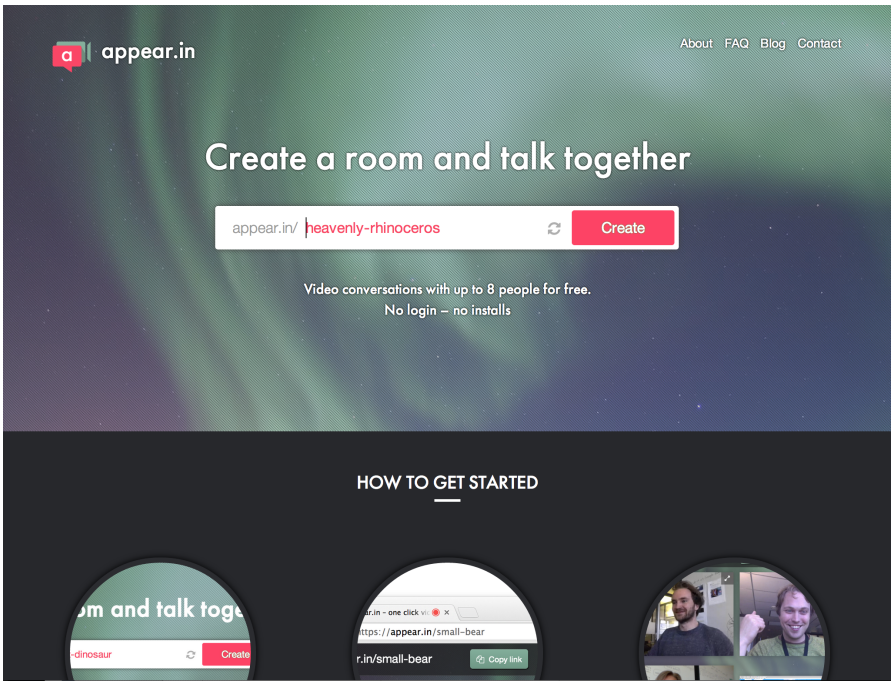
In a conversation
: The number of users going from the landing page and into a conversation. This entails that the person in question has understood the concept of how one establishes a conversation, which can be seen as the next step of user activation after entering a room.

The aggregate numbers are shown in tables 4.3 and 4.4.

Figures 4.3a and 4.3b show how the variations performed with respect to the two metrics, both with variation B plotted relative to variation A as the baseline.



(A) Variation A.



(B) Variation B.

FIGURE 4.2: Variations in the new landing page experiment.

As we can plainly see, variation B performs slightly better at moving people into rooms, but evidently fails to communicate the conversation concept, leading to a relative decrease in actual subsequent conversations.

While these numbers and plots set the scene, they are not very interesting from our

Variation	People	Conversions	Average conversion	Improvement	Certainty
A	10,100	7,946	78.67%	-	-
B	9,931	7,883	79.38%	0.90%	89.04%

TABLE 4.3: Comparison of "Visited room" conversion ratios for variations *A* and *B*.

Variation	People	Conversions	Average conversion	Improvement	Certainty
A	10,009	3,602	35.99%	-	-
B	10,172	3,781	37.17%	-3.29%	95.98%

TABLE 4.4: Comparison of "In a conversation" conversion ratios for variations *A* and *B*.

adaptation perspective. We are first and foremost interested in seeing whether these numbers vary significantly between segments of the user base, thus providing us with a predictive bias from which we can extrapolate our user adaptations.

Section 4.5 breaks the numbers down cluster by cluster, and investigates to what degree this information can be used to adapt the interface.

4.4 User clustering

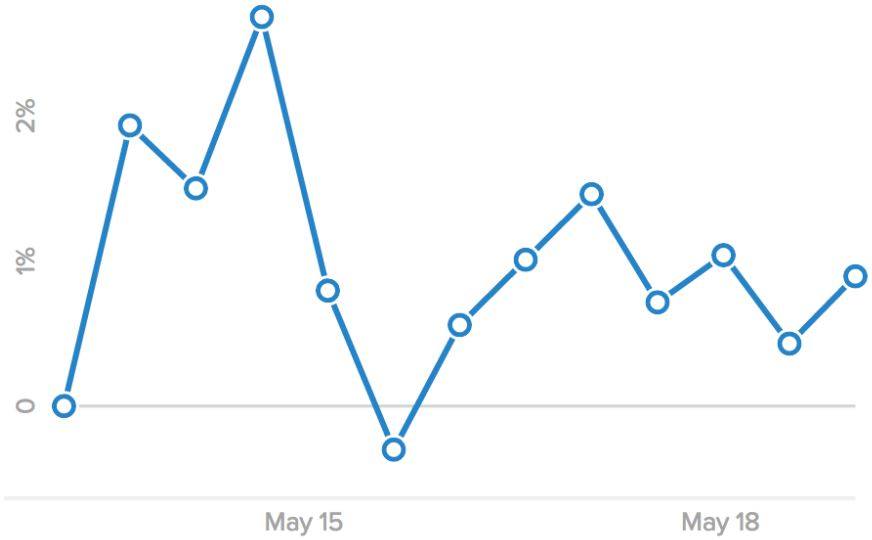
The clustering approach was iteratively improved on over several months. Large amounts of new data was gathered through the developments process, and with it the parameters were tuned and adapted.

Figure 4.4 shows a set of clustering runs. Data was timeboxed per month, from January thru May, and k-means was run for a range of k from 3 up to 10. The other parameters used, as well as the particular resulting cluster set from February, are discussed in further detail in section 4.4.3.

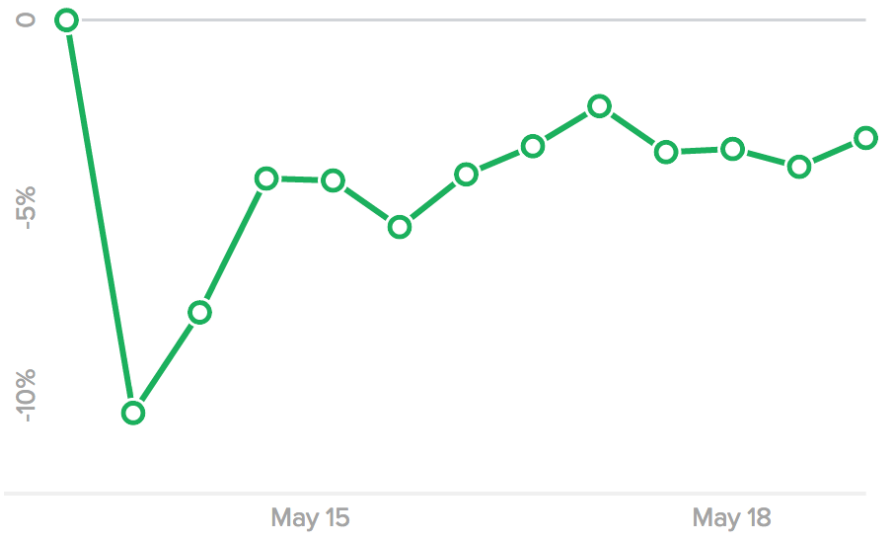
4.4.1 Varying demographics

Through the spring of 2014, appear.in received quite a bit of media attention, and different times saw user influx from a large variety of demographic origins.

The coverage varied from technical showcases and industry newsletters to the service being featured in Hungarian newspapers and on BBC World News. Moreover, these spurs of media attention seems to mostly have been independently initiated, and as a result we saw large peaks of visitors from quite specific locations at different times. Overall, though, the traffic was distributed quite evenly over a large number of geographical areas.



(A) Comparison of the variations by the "Visited room" metric.



(B) Comparison of the variations by the "In a conversation" metric.

FIGURE 4.3: Variation B performance, relative to variation A (the baseline).

The *timespan* parameter was used extensively to compare these, and aided in avoiding becoming subject to these various demographical biases.

4.4.2 Axis normalization

@TODO

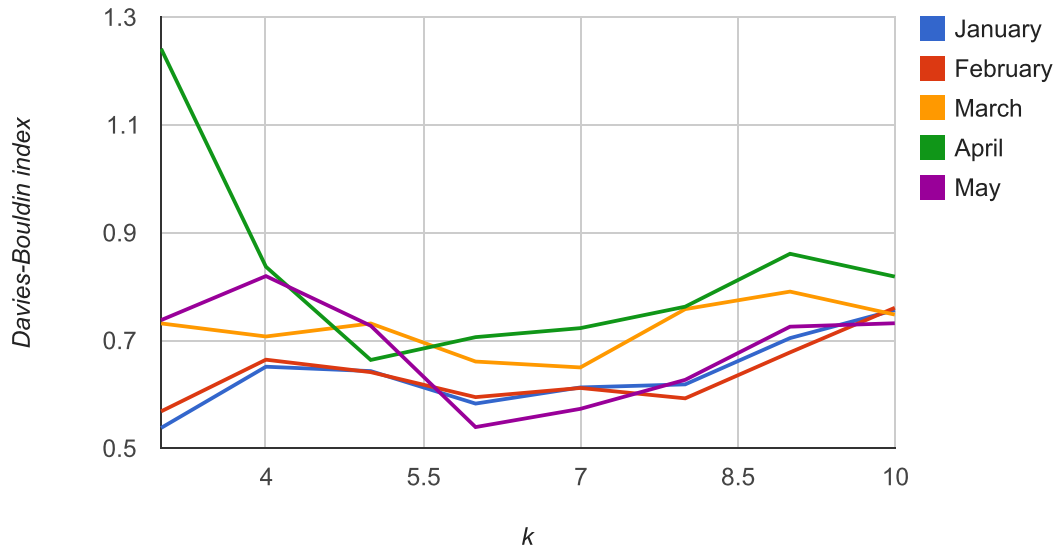


FIGURE 4.4: k vs. Davies-Bouldin index, for clusters with data for January to May.

4.4.3 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 2.3.2 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.

Although irrelevant to the experiment, a detailed analysis of the clusters depicted in figure 4.5 is available in A.1.

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.

The radar chart in figure 4.5 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 (clockwise around the chart):

1. chat messages sent

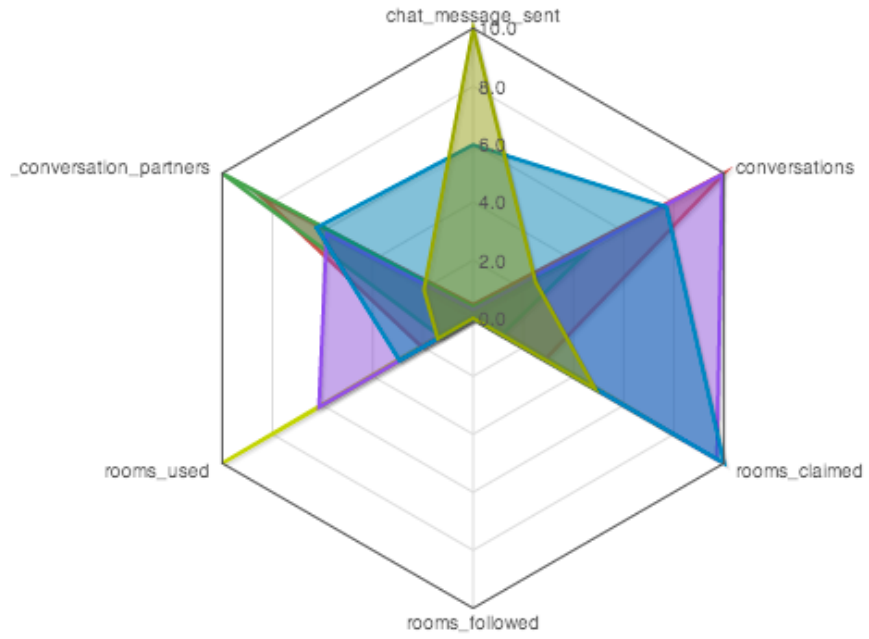


FIGURE 4.5: Radar chart comparing clusters generated from data for February 2014.

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.

4.4.4 The applicability of manual cluster analysis

When analyzing a set of clusters, it is tempting to focus on the extremes – the easily stereotyped user profiles – as we have done above. However, we see that a vast majority of users are indeed too close to the origo to be confidently labelled as some kind of stereotype.

Thus, although tempting from a business intelligence type of perspective, manual cluster stereotyping becomes a highly speculative exercise whose conclusions are extremely hard to confirm without demographic data readily available.

We will return to this issue in chapter 5.

4.5 Adapting the application

The task of adapting the application brings together previous A/B test results and the clusters discovered, as described in sections 4.3 and 4.4.

	off			on			total			
C	n	c	%	n	c	%	n	c	%	preferred
1	42	0	0.00	43	0	0.00	85	0	0.00	off
2	21	9	42.86	16	11	68.75	37	20	54.05	on
3	156	75	48.08	152	67	44.08	308	142	46.10	off
4	151	90	59.60	160	113	70.62	311	203	65.27	on
5	118	106	89.83	133	125	93.98	251	231	92.03	on
6	180	145	80.56	165	134	81.21	345	279	80.87	on

TABLE 4.5: Success of the different variations for each cluster C.

Table 4.5 shows the success of the different variations for each cluster. We see that there are indeed clear differences between them, and can therefore reasonably state that there exists an inductive bias on which predictions about future behavior can be based.

However, let us not jump to conclusions. In 4.3 we ran a controlled experiment with just over 20,000 people, and here we are seeing numbers totaling at just over 1300.

To illustrate the point of the next section, the clusters in table 4.5 were identified based on data from just *before* the A/B experiment was run. Cluster 1, in the first row, contains the same types of users as the ones described in A.1.1, namely users only hitting the front page.

As we can plainly see, the majority of the clustered users did not return to the service within the experiment time period, resulting in an extremely low relative participation rate. The other clusters however, comprised of more active users, see a much higher participation rate.

4.6 Identity persistence

The numbers in 4.5 tells us that it may not be enough to be able to predict what the current user base wants. What we *really* want is to use this information to improve the service for returning users. However, it is not enough to have predictive bias if we're unable to *recognize* our returning users. This section analyzes the impact of cookie impermanence.

As has been discussed extensively already, a major difficulty in adapting applications such as appear.in to its users is that there is no concrete notion of a user. Users are

anonymous, and the only way they are being tracked at all is through a random hash set in a tracking cookie (see section 2.2.3).

How, though, does this affect our adaptation efforts?

One way of gauging this is to look at the number of users we see returning. As an extreme, consider a user who does not keep cookies across sessions. Every time this user returns to the site, he will be treated as a new, previously unseen user. By extension, for how long do we see each tracking cookie return before we never see it again?

Table 4.6 shows the decline over time of the number of returning users, as compared to the user base of January, February and March.

Users from month	1 month	2 months	3 months	4 months	5 months
January	23.20%	5.10%	2.90%	1.90%	0.90%
February	20.20%	3.80%	2.10%	1.10%	-
March	23.20%	5.40%	2.90%	-	-

TABLE 4.6: The degree to which a monthly unique set of users are seen again after n months.

Two plots of these numbers can be seen in figure 4.7, with normal scale and logarithmic scale.

As we see, just over 20% are seen again the month after they first visit the site. Only about 20% of these users again make it through to the subsequent month. After three months, less than 3% remain in all three cases.

As a more general measure of this, take figure 4.6. The histogram shows the number of persons having visited the site at least twice, against the number of days separating their first and last recorded visit. As we see, it takes less than a month before the number of users has fallen an entire order of magnitude.

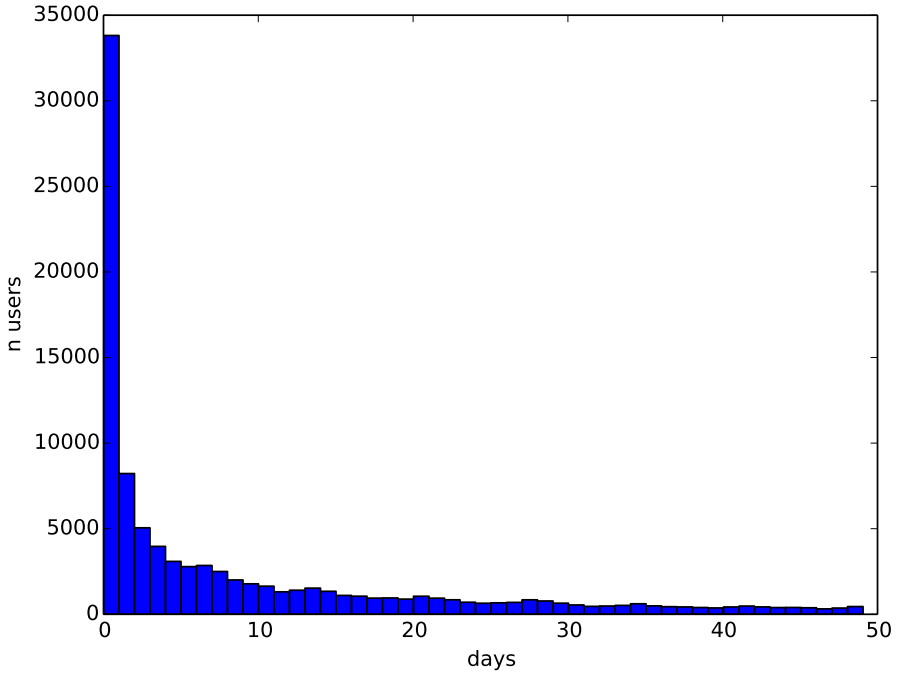
4.6.1 User retention factors

The perceived rate of retention is subject to three different factors:

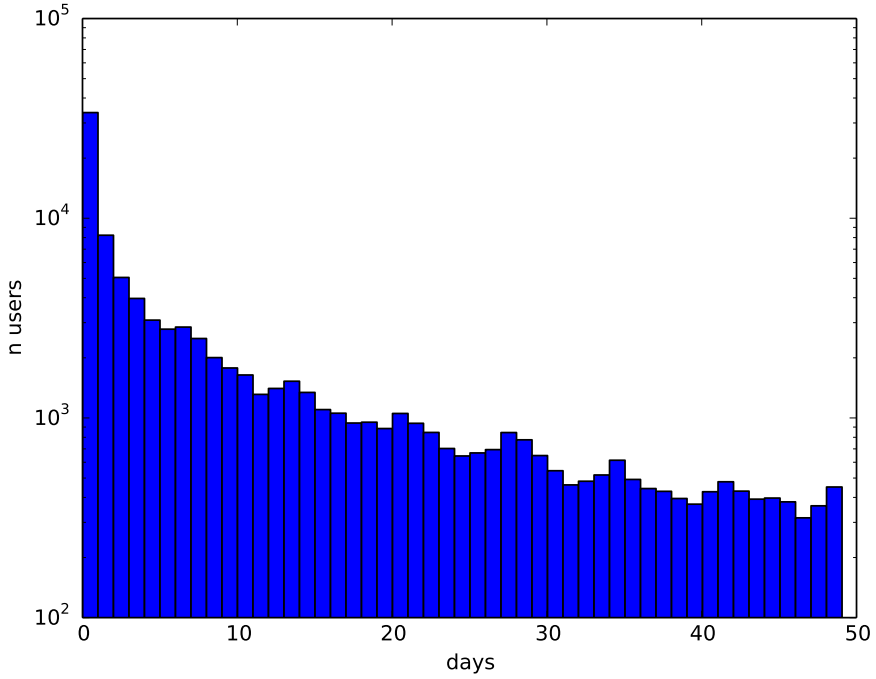
1. The user did not use the service again.
2. The user cleared the browser cookies.
3. The user changed browser or computer.

There is all reason to believe that we are looking at a combination between all three, yet hard to know how much each contributes to the trend. We naturally assume that most users perceived as not returning simply do not return.

However, the survey performed by Teltzrow and Kobsa [3], as discussed in section 2.2.4, states that more than half of all users clear their cookies periodically, and a significant amount of users reject cookies altogether. We will therefore assume that the last two factors also contribute in a significant degree to these numbers.

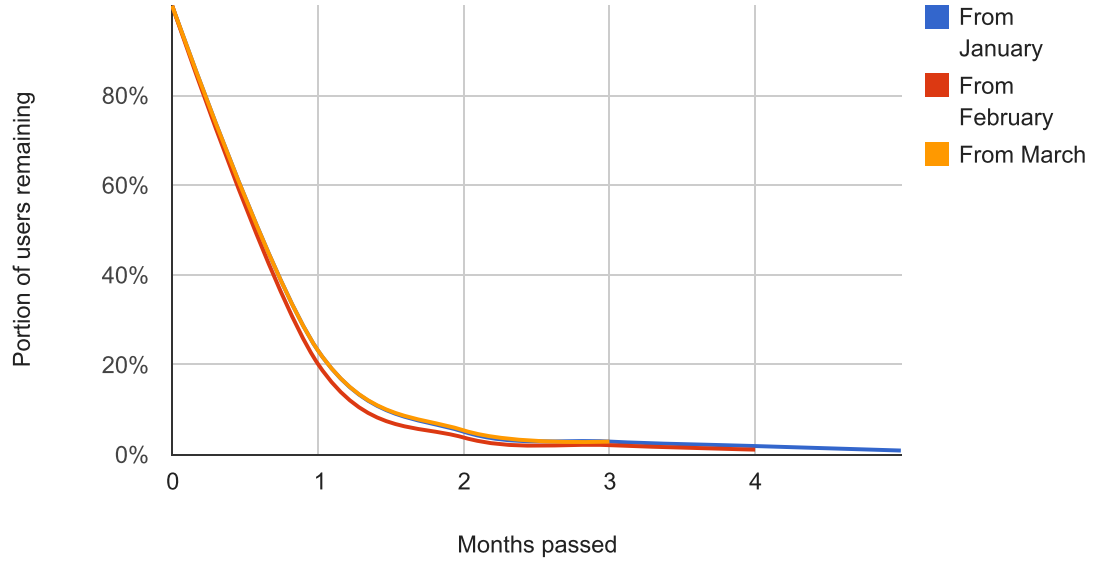


(A) Linear scale.

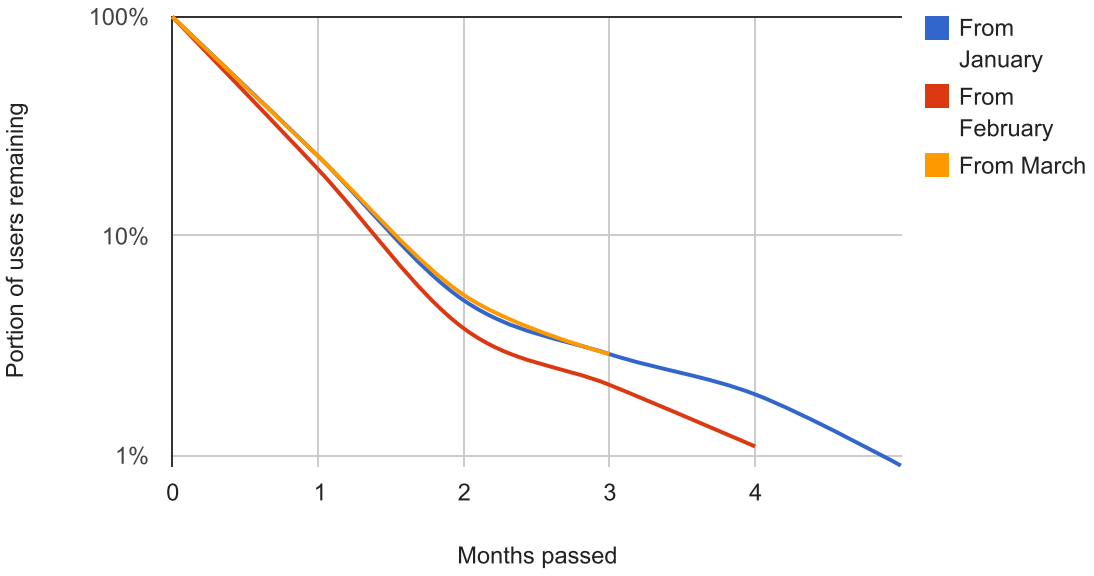


(B) Logarithmic scale.

FIGURE 4.6: Returning users' site usage time spans.



(A) Perceived user persistence.



(B) Perceived user persistence, logarithmic scale.

FIGURE 4.7: Perceived user persistence, illustrating the identified user dropoff rate.

Chapter 5

Conclusion

5.1 Summary and discussion

This thesis has presented a system capable of identifying user clusters, and the ways in which these differ from each other in their interactions with the application. The clustering results are clear and consistent over several months of largely varying user base demographics.

There is also evidence of clear differences in feature adoption across the identified clusters, presenting an inductive bias on which user adaptations can theoretically be based.

However, some issues presented themselves, particularly pertaining to the short life span of tracking cookies. As touched upon many times throughout the thesis, appear.in does not have any user identification mechanisms apart from that of identifiers stored in client-side cookies. As most users seem to clear their browser cookies regularly, we are only able to apply the adaptations to a small percentage of the users, as most users are perceived to disappear after a few weeks of activity.

This makes the scheme proposed in this project unfeasible in a production setting. The computing and data requirements arguably far outweigh the potential benefits, especially due to the relatively miniscule reach of the tentative user adaptations.

5.1.1 Generality of the results

The poor performance of the proposed system is mostly due the issue of unstable identity and cookie impermanence. In this regard, the results should generalize to any system whose user identification scheme relies solely upon browser cookies.

Indeed, the results herein confirm the position of the literature in that stable user identity is a necessary prerequisite for personalization.

5.2 Suggestions for further work

It would be interesting to determine the distribution of the user retention factors introduced in 4.6.1. Theoretically, albeit unlikely, the user dropoff perceived could simply all be due to users not returning to the site, their cookie clearing behavior not being a significant factor. To support the conclusions above, surveying users on this subject would be helpful.

On a related note, it would be interesting to see an up-to-date survey on users' attitudes towards first-party tracking cookies and online privacy, to see if they match the ones outlined in 2.2.4.

There are also several ways the approach taken in this project could be improved. The time window for determining adaptations turning out to be the biggest challenge of the suggested approach, it would be interesting to see the performance of an online scheme in comparison. Not requiring behavioral data to pass through the entire offline feedback loop in order to be taken into consideration, should remedy at least some of the worst symptoms of cookie impermanence. Moreover, as touched upon in 2.3, a collaborative filtering might be able to serve the purpose. It is, however, doubtful whether any online approach would alleviate the identity situation enough to be useful.

Another question of interest is: what kind of additional metadata would be needed to remove us from the problem of cookie impermanence? If provided with IP-addresses and the room names in use, for instance, would that be enough to enable consolidating user event streams – even across browsers and computers?

Appendix A

Evaluation Results

A.1 Clustering results

This section contains a comprehensive qualitative description of the 6 largest clusters identified among user models generated compiled from February data.

A.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.

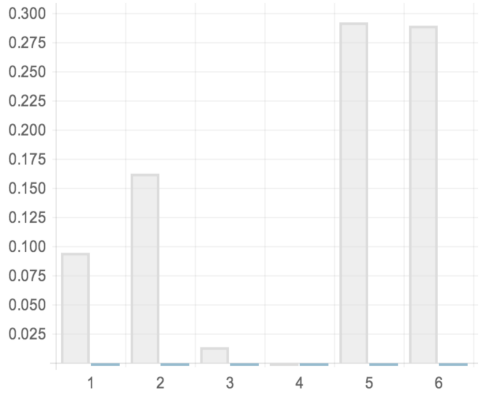
A.1.2 Cluster 2: Trying out the service alone

As shown in figure A.1b, the users in this cluster score close to 0 on every feature except the number of rooms used – most notably, the number of conversations.

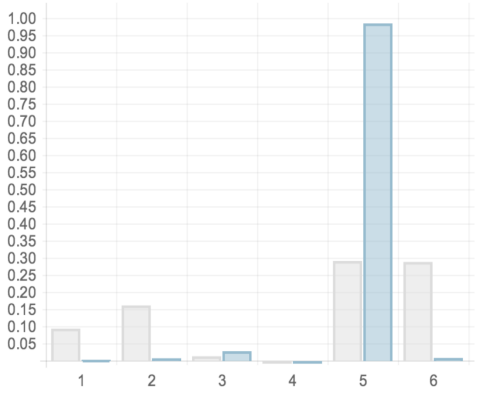
Persona 2. The user has tried out the service, but not ever conversed with another user.

A.1.3 Cluster 3: Simple users with small networks

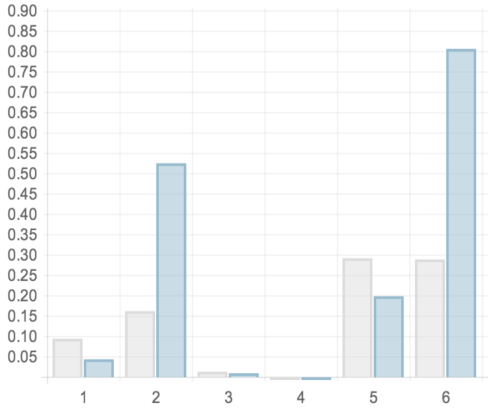
Users in cluster 3 (see figure A.1c) don't use any of the more advanced features of the service, like chatting, claiming or following a room, but on average they have taken part in just over 5 conversations.



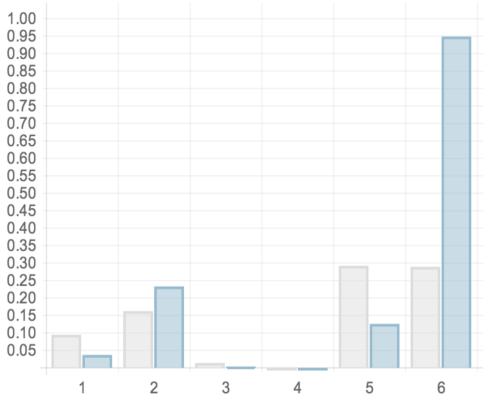
(A) Cluster 1: Front page hitting users



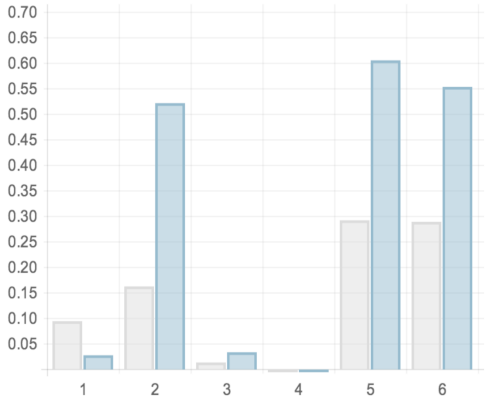
(B) Cluster 2: Users trying out the service alone



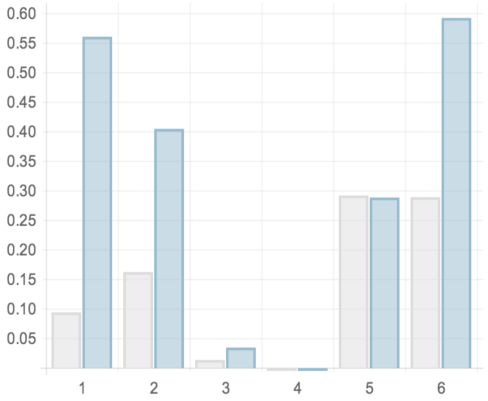
(C) Cluster 3: Simple users with small networks



(D) Cluster 4: Simple users with large networks



(E) Cluster 5: Incognito users



(F) Cluster 6: Chatty, simple users

FIGURE A.1: 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.

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

A.1.4 Cluster 4: Simple users with large networks

The users in cluster 4 (see figure A.1d) 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.

A.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 A.1e.

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

A.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 A.1f. 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.

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

Appendix B

Source Code

B.1 The k-means clustering algorithm

```
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
```

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] Alan Montgomery and Kannan Srinivasan. Learning About Customers Without Asking. *Tepper School of Business*, (324), 2002.

[3] 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.

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

[5] 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.

[6] Cyber Dialogue. Cyber dialogue survey data reveals lost revenue for retailers due to widespread consumer privacy concerns. *New York, Cyber Dialogue, November*, 7:2001, 2001.

[7] L Behrens. Privacy and security: The hidden growth strategy. *Gartner G2 Report*, 2001.

[8] S Fox, L Rainie, J Horrigan, A Lenhart, T Spooner, and C Carter. Trust and privacy online: Why americans want to rewrite the rules. pew internet & american life project, washington. *A Pew survey of*, 2, 2000.

[9] Mary J Culnan. The culnan-milne survey on consumers & online privacy notices: Summary of responses. In *Interagency Public Workshop: Get Noticed: Effective Financial Privacy Notices*, pages 47–54, 2001.

[10] Harris Interactive. A survey of consumer privacy attitudes and behaviors. *Rochester, NY*, 2000.

[11] Personalization Consortium et al. Personalization & privacy survey. *Los Angeles*, 2000.

[12] A. Ruiz-Martínez. A survey on solutions and main free tools for privacy enhancing Web communications. *Journal of Network and Computer Applications*, 2012.

[13] Nick Nikiforakis, Alexandros Kapravelos, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. Cookieless Monster: Exploring the Ecosystem of Web-Based Device Fingerprinting. *2013 IEEE Symposium on Security and Privacy*, pages 541–555, May 2013. doi: 10.1109/SP.2013.43. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6547132.

[14] Ove Sorensen and Christian-albrechts-universitat Kiel. Zombie-Cookies : Case Studies and Mitigation. pages 321–326, 2013.

[15] Van Eijk, Van Der Plas, and Van Der Sloot. Online tracking: Questioning the power of informed consent. 2011.

[16] P. Berkhin. A survey of clustering data mining techniques. In Jacob Kogan, Charles Nicholas, and Marc Teboulle, editors, *Grouping Multidimensional Data*, pages 25–71. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-28348-5. doi: 10.1007/3-540-28349-8_2. URL http://dx.doi.org/10.1007/3-540-28349-8_2.

[17] Edward W Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics*, 21:768–769, 1965.

[18] Michael JA Berry and Gordon Linoff. Data mining techniques for marketing, sales and customer support. john willey & sons. *Inc., 1997, 454 P*, 1996.

[19] Maria Halkidi. On Clustering Validation Techniques. pages 107–145, 2001.

[20] 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.

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

[22] 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.

[23] 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.

[24] 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.

[25] Wolfgang Pohl. Labour - machine learning for user modeling. In *Design of Computing Systems: Social and Ergonomic Considerations (Proceedings of the Seventh International Conference on Human-Computer Interaction), volume B*, pages 27–30. Elsevier Science, 1997.

[26] Wolfgang Pohl. Logic-Based Representation and Reasoning for User Modeling Shell Systems. *User Modeling and User-Adapted Interaction*, 9:217–282, 1999.

[27] Krzysztof Z. Gajos, Mary Czerwinski, Desney S. Tan, and Daniel S. Weld. Exploring the design space for adaptive graphical user interfaces. *Proceedings of the working conference on Advanced visual interfaces - AVI '06*, page 201, 2006. doi: 10.1145/1133265.1133306. URL http://portal.acm.org/citation.cfm?doid=1133265.1133306.

[28] Leah Findlater and Joanna McGrenere. Impact of screen size on performance, awareness, and user satisfaction with adaptive graphical user interfaces. *Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI '08*, page 1247, 2008. doi: 10.1145/1357054.1357249. URL http://portal.acm.org/citation.cfm?doid=1357054.1357249.

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

[30] Ron Kohavi, Randal M Henne, and Dan Sommerfield. Practical Guide to Controlled Experiments on the Web : Listen to Your Customers not to the HiPPO. 2007:1–9, 2007.

[31] Ron Kohavi, Roger Longbotham, Dan Sommerfield, and Randal M. Henne. Controlled experiments on the web: survey and practical guide. *Data Mining and Knowledge Discovery*, 18(1):140–181, July 2008. ISSN 1384-5810. doi: 10.1007/s10618-008-0114-1. URL http://link.springer.com/10.1007/s10618-008-0114-1.