ITN

Master of science and media technology

Collection and visualization of customer-specific data for the web

Gabriella Ivarsson
Oscar Persson

Supervisor:
Jan Petersson
Examiner:
Camilla Forsell

Norrköping
June 4, 2013

Abstract

As the Internet grows larger and the use of it expands, more people are developing for the web. This rapid development is making it more clear how important it is to think of the intended user. Twingly works with blog tools towards several different customers. Their data customers are interested in indexed blogs and the information that millions of blogs provide. The other type of customers are widget customers. They use Twingly's widget on their site to track what people are blogging about. For example, what products are popular at the moment in the blog world? Twingly wants to offer their customers the use of a dashboard with visualizations of customer-specific data. It would also be used to attract new customers. The purpose of this dashboard is to, in a user-friendly way, visualize the different datasets that are indexed by Twingly's system and are of interest to the customer. This should be done in real time on the web so that the customer always has the latest data. Sinatra (Ruby) is used in combination with Haml, PostgreSQL, Twitter Bootstrap and Rickshaw to create the dashboard itself and its graphs. Recovering and collecting the data is done in collaboration with Twingly's developers. An API is constantly pushing data to the dashboard with a set up stream. In order to make the dashboard faster, a PostgreSQL database is used to store some of the data to ensure faster page loads. The end result is a dashboard with good usability and a clean look. The graphs used to visualize the data are effective with interaction that all types of users can understand. Even if the graphs are basic, they give the user a lot of information. The legends used for the graphs are customized to work well on both a computer and a tablet. The time slider used for the graphs are also compatible with a tablet. The blog overview page gives the user a lot of information about new blog posts, what people are saying about them and what type of people are visiting their site. The use of the dashboard requires a log in and this enables the dashboard to be customer-specific. The data is continuously streamed to the dashboard, making it work in real-time. The dashboard is stable and every part of it has been carefully considered and thought through. The goals were reached and the end result is close to what was first planned in the beginning of the project. Twingly will launch it to the intended users in the near future as a part of their establishment at the international market.

Acknowledgements

We would like to thank Twingly for giving us the opportunity to do our Master's Thesis at their company. Thank you to the developers at Twingly who helped us along the way. We would also like to thank our supervisor Jan for being our guiding light through the jungle called report writing. We could not have done this without our examiner Camilla, thank you.

Finally, thank you to our families and friends for supporting us along the way.

Contents

1 Introduction

This report is the result of a Master Thesis Project, executed through the Master's of Science in Media Technology and Engineering Program at Campus Norrköping, Linköping University.

1.1 Problem description and background

The science of designing for human-machine interaction dates back to the early 1900s. Back then, it focused on the physical interaction between the machine and its operator's ability to handle it. [5] Since then, a lot has happened in the field of technology. The World Wide Web has made its breakthrough and web usability principles with it. As the Internet grows larger and the use of it expands, more and more people are developing for the web. This rapid development is making it more and more clear how important it is to think of the intended user. Is your user a teenager? Is it someone with a lot of computer skills? What will the user use the site for? In information visualization it is of uttermost importance to think of the usability. What is the best way to visualize large amounts of data? How will the user interact with the data? Since it is important for the application to be self-explanatory, it is necessary to use techniques that are natural and intuitive for the user.

Twingly is a data mining company from Linköping. It is a relatively young company founded in 2006. Their business strategy is to index as many blogs as possible and they strive towards having the best coverage, quality and support with their focus in Europe. Today, Twingly has stored data from over 62 million blogs in 30 languages, although these numbers are constantly growing, both from new blogs being created and from Twingly expanding their search areas. Currently, Twingly has over 100 customers in 16 countries. All the customers subscribe to some part of the indexed blogs and statistics, but each customer can be one of two types. The first type is *data customers*, see section 1.2.1, that subscribe to raw data concerning all blogs in the languages that they subscribe to. The second type of customer is a *widget customer*, see section 1.2.2. They use Twingly's widget on their site to track what people are blogging about them and encourage bloggers to link to and write about them. The majority of the customers are widget customers. Twingly wants to introduce a new tool for making the data more accessible and valuable to their customers. They wish to offer their customers the use of a dashboard, see section 2.1, with visualizations of customer-specific data. This data was previously sent as graphs in emails to the customers, so a dashboard where the customer could interact with the data in real-time, would really add to their business value. The dashboard would also be used to attract new customers, since it could show, in an easy and intuitive way, the data/service they could buy. This dashboard will be a major part in order for Twingly to establish itself on the international market. Twingly will wait to launch into a bigger international market until the dashboard is completed.

1.2 The users

Twingly offers a variety of products that allow the customer to interact with the world of blogs. They have two main types of customers, data and widget. The dashboard will be developed for both kinds of users with different views as they sign in to the dashboard. Since the user can be any employee at the customer firm, it can be assumed that the computer skills of the users will vary. Thus, the design must be kept simple and clean to remain understandable. It is important to show the intended use of the dashboard and make it intuitively clear what its purpose is. This is all important. The simple and clean design will draw the user in, tempting him/her to use the dashboard the first few times. Animations can also help to draw in the user and increase the experience. However, the dashboard also needs to be useful to make the user want to return and use it again.

1.2.1 Data customers

Data customers want to now what is talked about in all blogs. They can use data from blogs to find what has been said about them, their clients and competitors. This is a service that the customer subscribes to and can include a number of languages. It can also be used to see trends in bloggers activities, etc. See an example of an indexed blog in figure 1.1.

Figure 1.1: Example of blog that is indexed by Twingly.

1.2.2 Widget customers

Twingly also offers a widget. This widget is used by online shops, magazines, etc. to track what bloggers are linking to on their site. Then the widget customers can read the context of the links, follow up marketing events and contact bloggers. This data is useful for this type of customer, since they can use it to plan their marketing strategy in a better and more effective way. It can also help an online shop in seeing what is popular so that they can plan how to stock. An example of a web site using Twingly's widget can be seen in figures 1.2 and 1.3. The data retrieved using these widgets is what will be displayed in the dashboard.

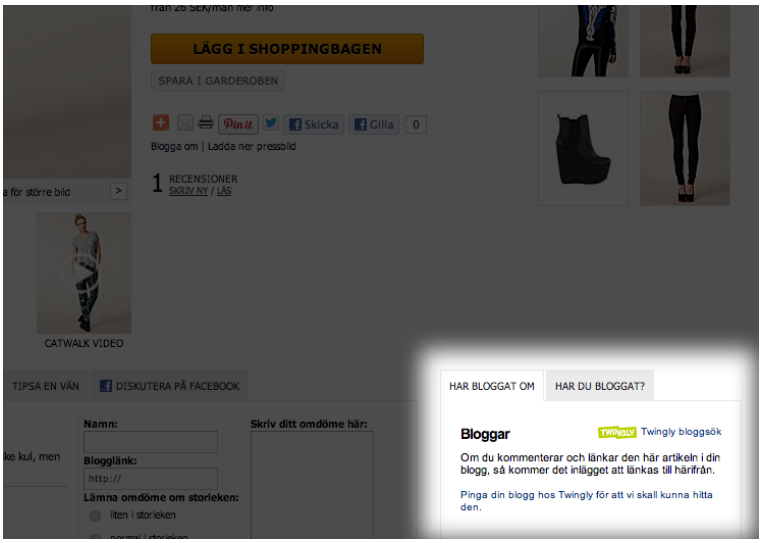

Figure 1.2: Example of a webshop widget customer.

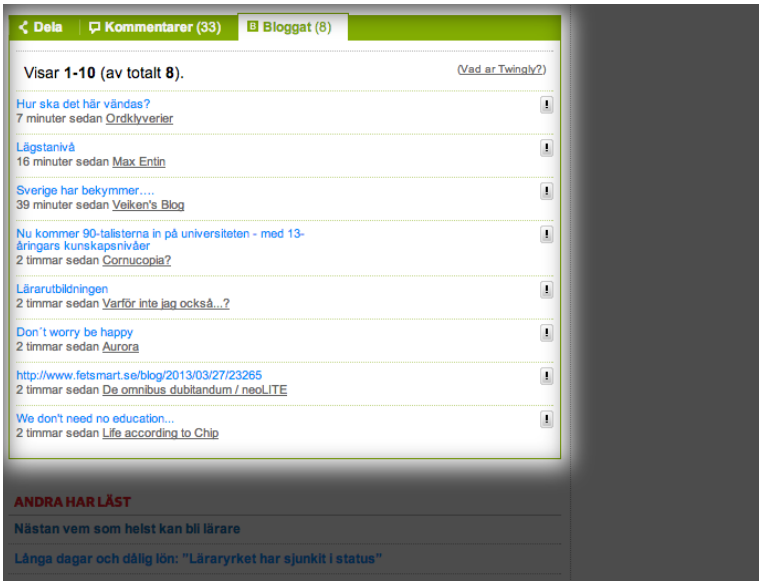

Figure 1.3: Example of a newspaper widget customer.

1.3 Objectives

The goal with this Master's Thesis was to create a dashboard for Twingly's customers. The purpose of this dashboard is to, in a *user-friendly* way, visualize the different data sets that are indexed by Twingly's system and are of interest to the customer. The dashboard has to be user-specific and have a secure log in for the users. The data represented should be in *real-time* to allow for a more interactive design. Real time updates will make the dashboard even more useful for the customer. The GUI should be in line with Twingly's graphical profile, as well as introduce new creative ways of navigation. Another goal is to make the dashboard work well on tablet if time allows it. Smartphones are excluded due to their small screen size, which would require an entirely different layout. It should work on a tablet so that Twingly and their customers can view the data in a simple way. This requirement introduces a new aspect to the design. For example, the design has to adapt to the available screen size and the sliders have to work on a touch screen. It is also important that the graphs are as big as possible, as it has to be easy to read the data.

1.4 Methodology

A low-fi prototype is made from paper and used to simulate navigation inside the dashboard. The mentors at Twingly will then give feedback and a second prototype will be made based on the results. To start off, the conditions of the project have to be investigated. After this, the programming languages and frameworks will be decided upon. These are described further in section 2.4. Both previous skills and what is best suited for the project is weighed into this decision. Since the supervisors at Twingly are familiar with many techniques and frameworks, their suggestions are of great value. The development environment first needs to be set up or updated. This includes installing the different technologies and frameworks used, see section 2.4. When this is done, the development of the project will begin.

Visualizations obviously need some kind of data to visualize. The data needed for the visualizations in the dashboard are not readily available. The data are spread through several different databases and needs to be retrieved and collected. This is done in collaboration with Twingly's developers. The data is then streamed to the dashboard using an API, as well as collected directly from databases. This data is owned by Twingly. One of the most sought after diagrams is the one with new blog posts over time. The first thing to create will be this diagram. It is important to get at least one diagram done early on. This is to know which data formats to use for the input data. The other developers at Twingly are then able to assist in getting the right data from the right sources. After this, the basics of the application are created. The application is first built using dummy data, to get the basic structure. When more data is available, dummy data will be replaced with actual statical data. The next step is to get data in real time . The development will be will made iterative and incremental in an agile fashion, but no specific development method will be used. However, the term backlog from Scrum will be used to refer to the list of tasks and features waiting to be developed.

1.5 Delimitations

Since the dashboard will be launched and available to Twingly's customers, the goal is for the dashboard to be as close to a finished product as possible. Some of the features that Twingly wishes to implement, would require data that Twingly currently do not have readily available. Some parts of the dashboard will therefore be implemented with real data, while some parts will use dummy data. Developing for the dashboard to

be used both on a tablet and a computer also comes with some restrictions. It will be possible to use the dashboard on a tablet, but not on a smartphone. This is due to screen size. The smaller screen on a smartphone makes it impossible to read the graphs on the dashboard. A version for smartphones would require an entirely different design.

1.6 Report structure

This report is organized as follows. Chapter 2 contains the theory of the thesis. It has an introduction to the theories, methodologies and technologies used. The feasibility study is explained in chapter 3 and discusses the work done prior to implementation. A description of the work done in this Master's Thesis can be found in chapter 4. The results are found in chapter 5 and the discussion in chapter 6. The last chapters are the conclusion (chapter 7) and future work (chapter 8).

2 Theory

To be able to create a dashboard with good usability and aesthetics, the underlying theory is important. The knowledge of what to do and why will ensure a good end result.

2.1 Dashboard

A dashboard is a tool used by a business to display a set of relevant information to a business user. The data is often displayed in real-time after collection from many different data sources. A dashboard should also be interactive to allow the user to view specific parts of the data. It is a way to display large amounts of data in an intuitive way and simultaneously save employee's time. [25] A good dashboard, according to Hetherington [25]:

- *Communicates with clarity; quickly, and compellingly. Simplicity is key.*
- *Has minimal unnecessary distractions, no matter how interesting, which could cause confusion.*
- *Organizes business information to support meaning and usability*
- *Applies the latest understanding of human visual perception to the visual presentation of information*
- *Is pleasant to look at*

2.2 Information visualization

Information visualization is the study of how to reinforce human cognition by using visual representations of abstract data in an interactive way. [6]

2.2.1 What is information visualization?

When trying to understand something, we try to make ideas clear and in focus, as well as arrange them. The fact that humans use visual metaphors when describing cognitive processes shows the relationship between what we see and what we think. One important part of these external aids are graphical invertions. The use of these has a long history, but the evolution of computers has made it possible to improve rendering and real-time interactivity. [6] Today, there is a lot of abstract information available and the complexity of this data is constantly growing. It is therefore useful to be able to visualize abstract data in a good way. [6]

Information visualization is the use of computer-supported interactive visual representations of abstract data to amplify cognition.[6]

The purpose of using visualizations is to give insight, not to show images. The goal of this insight is to discover, make decisions and get explanations. Information visualization helps to perform these tasks. [6]

2.2.2 Why use information visualization?

Larkin and Simon [13] conducted a study that shows how visualizations amplify cognition. Their conclusions were that by grouping information that is used together, a lot of searching was avoided. It also showed that using location to group information of one element reduced search and working memory. The third thing the study showed was that visual representations automatically support many perceptual interfaces which are easy for humans. This will reduce the cost of some operations. [13]

2.2.3 How to visualize

There are many ways to visualize data. There are the basic bar charts and circle diagrams and, if the data is multi-variate, there are more types such as scatter plots, table lenses and parallel coordinates. However, this data is mostly tied to the dimension of time. The most used graphic arrangement in newspapers and magazines over the world are time-series plots. The design has one dimension with the regular rhythm of time. It is in big data sets with varying data that time-series graphics shows its true power. Simple linear changes will not be interesting in this design, as it can be summarized in numbers. [24] Time-series graphics have been appearing in scientific writings since the 1700s and the first one ever discovered dates back to the 10th or 11th century [24]. This supports the theory, that Tufte [24] himself promotes, that this is the simplest graphic design for interpreting time-oriented data. Tufte explains how *"graphics must not quote data out of context"* [24]. In order for the graphics to be truthful and revealing, the data must be comparable to its context. If a graph only includes some data for a specific year and the hidden span of the available data is 20 years, then the graphics are leaving important questions unanswered and there is nothing to compare with. This is one way for graphics to lie and deceive, but the same goes for any form of communication. [24] The essential purpose of statistical visualizations are to help people reason about quantitative information. These visualizations should draw the attention of the viewer and make them see the sense and substance of the data.

2.2.4 Data types

In everyday life, the need for human insight into large amounts of data is common. There are three different types of data that can be visualized. Most common is numerical data. Data could also be ordinal (naturally ordered) or categorical (names, etc.).[19] The data used in the dashboard developed will mostly be numerical. Examples of data that are not numerical in the dashboard are the links and texts.

2.2.5 Computation

There are many issues associated with information visualization, all worsened by using computers.[19] Some of these are:

- **Selection** The first thing to do when visualizing large amounts of data is to select the part of the data that relevant to your visualization. What information will be useful in the visualization? Can the selection be done automatically? [19]

- **Representation** The next step is to decide which way to represent the data. There are many different methods, but which is the best for this visualization? Maybe a combination? [19]
- **Presentation** Since the screen size is often limited, it is of extra importance to organize the data in some way. [19]
- **Scale and dimensionality** Large amounts of data with many dimensions needs to be scaled. It is important to know how the scale influences the design of the visualization tools. [19]
- **Rearrangement, interaction and exploration** One of the key aspects for the viewer is to be able to interact with the data. If the data is rearranged, the visualization often gives additional insight. In general, interaction with the data enhances the tool's effectiveness. [19]
- **Externalization** As stated in the beginning of section 2.2.1, visualization is the internal model of the viewer. What the viewer usually sees (on a computer screen) is called the externalization of the data. [19]
- **Mental models** Understanding the internal model of the viewer makes the design of visualization tools easier. [19]
- **Invention, experience and skill** All visualization tools have been invented by someone and the success of the tool relies on the designer's knowledge and understanding. [19]

2.3 Usability

Since the dashboard will be used by Twingly's customers, who may not have advanced computer skills, it is of extra importance to think of the usability of the dashboard.

2.3.1 What is usability?

According to Krug [22] the essential thing in web usability is:

> *Don't make me think!*

A web page should be obvious and self-explanatory. When developing a site, the challenge is to get rid of all the question marks. When using a web page, the user does not read the page, the user scans it. This means that it is important to have a clear structure.[22] First impressions are important in real life, but even more important on the World Wide Web because a web page cannot respond and modify itself when visitors react negatively. One of the key principles in web usability, according to Badre [5], is consistency, since humans learn skills through habit. How easily the user can navigate and perform a task is called coherence, also an important aspect of web usability. As in designing printed material, a good way to create the layout of the page is to think of the negative space, i.e. white space. Start off with the thought of preserving all white spaces and then filling as little of it as possible. This should be done while still accomplishing the purpose of the site. [5] Color is another way to organize a page. Black and white should be used with another color as a supplement. The convention is to use a maximum of six colors on a web page. [5] In web development, it is important to think of the compatibility between different devices and browsers that might be used for the site. Controlled navigation and the cost of switching between pages should be considered. There are multiple points of entry to a website, which lead to problems if not considered. It is also important to have good usability due to the ease of which the user could switch to another web page that works better. [5]

2.3.2 The User

When designing a web page, it is important to define the user. There are both group differences (language, culture) and individual differences (skill, personality). Identifying these differences will allow for the right design and interaction style. [5] Some people suggest that designing for all different types of users requires lowest-common-denominator strategies, i.e. making the system as simple as possible. Schneiderman and Plaisant [20], however, have the experience that rethinking a design for different users and situations often results in a better solution for all users. One example of this from the real world is the adaptation of sidewalk edges for wheelchair-users. This adaptation also benefits parents with strollers, people with luggage, etc.[20] In designing interactive systems it is important to understand the perceptual and cognitive abilities of the intended users. The fact that humans rapidly can interpret sensory input and begin complex actions makes computer systems possible. Memory, decision making, learning, knowledge, hunger, personality, etc. all have great influence on the user's interaction with the system. [20]

2.3.3 Eight golden rules of interface design

1. *Strive for consistency.* This rule is the one that is most often violated. There are many types of consistency, for example that consistent sequences of actions are used in similar situations, the same terminology is used in the same context, etc.

2. *Cater to universal usability.* To see the difference in users, and adapting the interface for both experts and novice users, is important.

3. *Offer informative feedback.* For every action from the user, there should be a responding feedback.

4. *Design dialogs to yield closure.* Sequences of actions should be grouped together to give the user a sense of completion and accomplishment. One example of this is the different pages when buying something on an e-commerce website. Selecting the products is followed by a checkout page and then a payment page.

5. *Prevent errors.* A system should be designed so that the user cannot make large errors. If an error occurs, it should be detected and give simple instructions of how to recover.

6. *Permit easy reversal of actions.* Actions should be reversible if possible. This will calm the user since errors can be undone.

7. *Support internal locus of control.* The user wants to feel like he/she is in control of the system. An inability to produce the desired results will give the user anxiety and dissatisfaction.

8. *Reduce short-term memory load.* Human information processing in short-term memory allows users to remember "seven plus or minus two chunks", chunks being for example menu options, at the same time. The display should therefore be kept simple. In this context, chunks means a blocks or parts of a layout.

A common reference is [20].

2.3.4 Graphs

It is of importance that graphs are interactive and simple to use. Since the dashboard that is to be created will be used on tablets and computers, the graphs have to work in an environment with touch interface.

When displaying graphs of large amounts of data, it is important to think of the usability. One important aspect of the graphs is how to control the input data. Since a lot of the data for the dashboard will be temporal, it is important to think of how to control the time span. How do you select a time interval? Should the languages be controlled for the entire page or for each individual graph? How much information should be displayed? When displaying large amounts of data, the graph could become jagged. It is therefore important to present the user with the ability to smooth the graph. Smoothing the graph will give it a more uniform appearance, which might help the user in drawing a general conclusion about the data.

2.3.5 Navigation

Navigation has the purpose of directing the user between different contents. Without it, the user would have to enter the URL themselves. [26] Christian Holst and Jamie Appleseed [26] explain how there are three types of navigation:

Formal navigation Links which are persistent, i.e. menus and search fields.

Meta navigation Contextual links and dynamic filtering.

Inline navigation Integrated links in the body content.

If the website has a lot of content then all the above navigations might be needed, but limiting navigation to one primary navigation will make for a simple and more restricted form of navigation. It is important that the top-level menu names are descriptive to the content that they hold. A user will not look or search for content that they do not know exists. When navigating, the user should be hinted to what happens next. Interaction can motivate the user to want to find out what that is. Animations that respond to user's performed actions will encourage them to continue. [26] It is important to have clear rules for navigation since this might be a difficult thing for the user. The National Cancer Institute [11] has put together some guidelines for designing navigation. These guidelines include some which apply to this implementation:

- *Standardize task sequences.* Allow users to perform tasks in the same sequence and manner across similar conditions.
- *Ensure that embedded links are descriptive.* When using embedded links, the link text should accurately describe the link's destination.
- *Use unique and descriptive headings.* Use headings that are unique from one another and conceptually related to the content they describe.
- *Use check boxes for binary choices.* Provide a check box control for users to make a choice between two clearly distinguishable states.

A common reference is [11].

2.4 Technologies

As a designer and developer it is important to think about the wide range of software and hardware platforms. Newer machines might have bigger storage capabilities, faster processors, etc. That which works well on one computer might not work on another. [20] The fact that the dashboard will be used on a tablet sets minimum requirements on the client side. On tablets, the screen size, likewise the performance, is limited. In developing the dashboard, testing different operating systems and web browsers is especially important. The dashboard will work on most major browsers but not on Internet Explorer (IE). This because some of the technologies used, such as the EventStream, are not supported by the latest IE. The fact that the dashboard is relatively fast will be important. It is important to do as much of the work as possible on the server side, to make the client side as responsive to user input as possible. How the technologies were chosen is discussed further in section 3.2. A diagram of the application structure can be seen in figure 2.1.

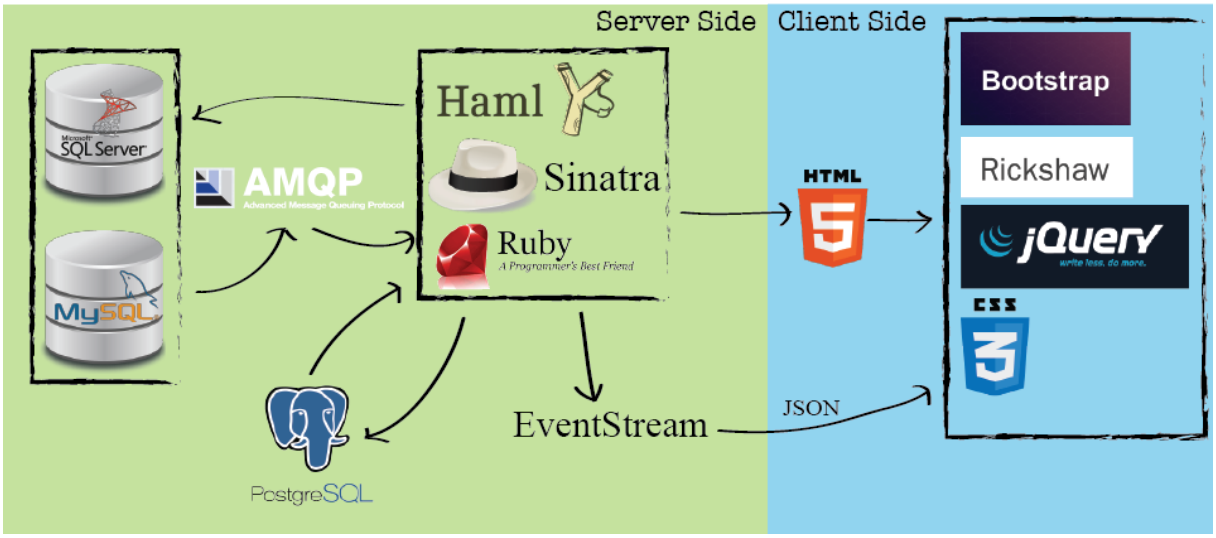

Figure 2.1: Diagram of the application structure.

2.4.1 Server side

Since it is important to do most of the work on the server side, the choice of technologies will be vital.

2.4.1.1 AMQP

Advanced Message Queuing Protocol (AMQP) is an open standard application layer protocol for message-oriented middleware [16]. AMQP is used to stream data from the databases to the dashboard. Using AMQP enables the dashboard to work in real-time. It is a technology that Twingly is already using and know well.

2.4.1.2 SQLite

SQLite is a software library that implements a self-contained and serverless SQL database engine.[18] Another technology, PostgreSQL (2.4.1.3), is introduced half way through the project. In the beginning, SQLite is

used for the storage of states. When deploying the dashboard, it is made clear that a more powerful and faster database is needed, so PostgreSQL is chosen as a replacement.

2.4.1.3 PostgreSQL

PostgreSQL is an open source object-relational database system. As it has been developed over many years, it is strongly reliable, with data integrity and correctness. It runs on all major operating systems. [8] On the dashboard, the streamed data has to be stored temporarily to allow the graphs to load faster on refresh. Once the data has been received by the graphs once, it will be saved in the PostgreSQL database and will load faster the next time.

2.4.1.4 Haml

Haml is a markup language and is used to simplify the HTML code. In contrast with other markup languages, haml has few symbols and is therefore simple to read and write. Haml is often used in combination with Sinatra, which will give a lot of benefits when developing and running into problems. [23]

2.4.1.5 Ruby

Ruby is an open source programming language. It has a simple syntax, making it easier to read and write, and is purely object-oriented. All of this makes it simple to use. [29] Since Ruby is so easy to write and read, using it was one of the first decisions made during the project. Previous knowledge of the language is also a positive thing, since the start-up time of the programming will be less.

2.4.1.6 Sinatra

Sinatra is a web development programming language in Ruby. It is often used for smaller applications with lower number of end points. Sinatra lets you do a lot with little. Sinatra is scalable and responsive, which was one of the goals with the development. [9] Sinatra is a good choice, since speed is important and the project will not need a complex structure. It will also work well together with the other technologies chosen. Sinatra has a "VC" (View-Controller) structure, i.e. the views are separated from the controllers. A MVC (Model-View-Controller) structure was desired, so Sequel [3] was picked as a database framework to add the model part to the structure.

2.4.1.7 Thin

Thin is a Ruby web server that handles communication between server and client. Thin evolved from the older web servers WEBrick and Mongrel and uses code from Mongrel to parse HTTP. Unlike these servers, Thin has better network I/O performance because it uses Event Machine for handling event network communications. [9] It is fast, stable, secure and easy to use. [14]

2.4.2 Client side

The client side should be fast and stable, to ensure a good user experience.

2.4.2.1 HTML5 and CSS3

Hypertext Markup Language (HTML) is the publishing language of the World Wide Web [27], where HTML5 is the latest "version". The Web Hypertext Application Technology Working Group (WHATWG) actually does not call it HTML5, but "HTML: The Living Standard" (the 5 was dropped). Thus, HTML is a version-less technology. Different browsers cherry-pick the feature they want to support and thereby features support varies. HTML5 includes new feature recommendations such as new semantic tags and client-side storage. [7] Cascading Style Sheets (CSS) is a simple way of adding format to a website. It is used to describe the formatting of a document written in a markup language, such as HTML5. [28]

2.4.2.2 Twitter Bootstrap

Twitter Bootstrap is a Front-end framework for HTML and CSS design templates. It also contains optional JavaScript extensions. Twitter Bootstrap provides widely used UI pattern standards. [17] Since a general and simple look is desired for the dashboard, Twitter Bootstrap is a good choice as you get a good basic structure.

2.4.2.3 D3.js

D3.js is a Javascript library used to manipulate documents based on data. It uses HTML, SVG and CSS to create graphs and emphasises on web standards. This gives full capabilities of modern browsers. [15]

2.4.2.4 Rickshaw

Rickshaw is built on top of D3.js and will be used to build graphs. It contains a collection of simpler graphs. [10] The graphs used in the dashboard are so called "basic graphs" in D3.js. Rickshaws range of simpler implementations should therefore be enough. With Rickshaw, less code has to be written to generate a result.

2.4.2.5 JQuery

jQuery is a fast, small, and feature-rich JavaScript library. It works in multiple browsers and makes things like Ajax much simpler to use. [12] As a time slider for the different graphs, JQDateRangeSlider was used. This slider, in contrary to Rickshaw's slider, allows the user to select ranges of dates and use this range to move in the graph. It also works on a touch screen, which is important in the dashboard. [2]

2.4.3 Testing

Testing was performed during the development of the dashboard. Functionality tests was done during the programming, both by the developers and the mentors at Twingly. Performance testing was carried out after each finished task and usability testing through out the development process. Usability testing ensures that technologies used and design decisions made are chosen on valid grounds from a users point of view. Otherwise it is easy for the developer to think that what he/she is doing is user friendly, when in actuality, it is not. The purpose of functionality and performance testing was to make sure that the dashboard is fast and stable. All testing ensures that the finished product is as good as possible.

3 | Feasibility study

At the beginning of the project, it was important to get a knowledge of what should be developed. This included getting to know the intended user and deciding upon frameworks and technologies to be used.

3.1 Prototype

Before writing any code, a low-fi prototype was made from paper and used to simulate navigation inside the dashboard. The first prototype was based upon the requirements given by Twingly, as well as theory on usability and information visualization. Keeping the GUI simple and intuitive was important since the user might not have that much previous experience. The colors chosen were based upon Twingly's graphical profile in combination with the directions about colors given by Badre [5] in section **??**. When the first prototype was finished there was a meeting with Twingly discussing the basic layout and functionality. The Twingly team gave feedback and suggested some changes. Since they know the intended user better and know what kind of data they usually ask for, this input was much appreciated. For example, it was learned that the graph visualizing new blog posts over time is one of the most asked after graphs. The dimensions and resolutions of time to use were also discussed during these meetings. The conclusion of those discussions was that this should be decided when viewing the graphs later on. It is important to give the data justice and display it in an acceptable way. This because of Twingly's wish to use the dashboard to attract new customers, as discussed in section 1.1. A second prototype was made and a group discussion held. Some minor changes were suggested and a basic layout was decided upon. The prototype served the purpose of providing a common look and feel for the dashboard early on before writing any code. This way, some of the obvious mistakes could be avoided. It was also a great opportunity to get feedback on the layout. See appendix 8 for images of the prototypes.

3.2 Frameworks

The mentors at Twingly suggested some technologies that could be worth investigating. Different frameworks, languages and environments were examined to find the ones most suited for the project, see section 2.4 for more information. Sinatra was chosen as the main language for the dashboard, based upon the following reasons. The developers at Twingly had previous experience of the language and thought that it would suit this project. It is also a Ruby-language and therefore previous knowledge in Ruby on Rails would make the start up time shorter. Sinatra is a lightweight language that lets you do a lot with little and since the dashboard will have a basic structure, it seemed fitting. It is unnecessary to write everything for every graph, so therefore a framework for making the composition of the graphs a simpler task is used. An earlier project

at Twingly had used a framework called D3.js. Evaluating this led to the discovery of another framework built on top of D3.js, called Rickshaw. Rickshaw contains elements to build from and allows you to write less with faster results. It was appropriate to use a JavaScript framework since writing plain JavaScript is time consuming and can make the code seem messy. jQuery is used since it is well known and has great documentation. jQuery's slogan is *"write less, do more"* which suits this project perfectly.

3.3 Conclusion

Each decision made during the development was of importance to the end result. The development of the dashboard's GUI was an iterative process that will take some time. The frameworks chosen were important to the speed and structure of the dashboard. It is possible to visualize the data in a simple and clear way using the technologies chosen.

4 The Work

A thorough investigation of the theory and conditions is important and gives a solid ground for the project. When this has been completed, the implementation can begin.

4.1 Data Retrieval

One thing that is a constant consideration during the development of the dashboard is:

How and where to get the data?

There are many obstacles to overcome during the development. The data is scattered throughout Twingly's different databases and servers. Even if the right database is found, the data is distributed in different tables and the keys are oddly named. To further complicate the data retrieval, some of the data needed is not readily available and have to be calculated. Retrieving and collecting all the data is done in collaboration with Twingly's developers. An API is constantly pushing data to the dashboard which sets up a stream, see figure 2.1. To make the dashboard faster, a PostgreSQL database, see section 2.4.1.3, is used to store some of the data prior to forwarding it to the view displaying the graphs. This way, some parts of the dashboard will pull its data from the PostgreSQL database, thus significantly reducing the loading time upon page loads. This occurs since there is no need to wait for the streamed data. In other parts of the dashboard the stream is used directly to do real time updates.

For the data to be real-time, the dashboard has to be notified when there is new data available. This is accomplished by opening an EventStream that pushes data as it receives it. The Advanced Message Queuing Protocol (AMQP), see section 2.4.1.1, is used to stream data from the databases to the dashboard. Using AMQP enables the dashboard to work in real-time. In AMQP routes are set up with different bindings. This allows for different data to be sent on the same stream using different bindings. In addition to AMQP, some data is collected directly from databases. In some parts of the application, an id is sent using AMQP and the data is then collected from the database using this id. There are many different databases used in combination with each other. One is a MMSQL database, another is a MySQL. In addition to this, a PostgreSQL database is created to be used in saving some of the data temporarily, as discussed previously. Even if all of these sources of information are used, all data needed is not readily available. Some of the data needs to be calculated. Other data will only be available in the future when Twingly adds the collection of it to their tools. One example of this is the product information on the widget page. Today, it is impossible for Twingly to know if an inlink, a link made to one of the customers pages, is a product (dress, shoes, etc.) or simply a link to a newspaper article. This can only be known if the identification is added to their widgets.

4.2 Graphs

The Rickshaw library, see section 2.4.2.4, is used for generating graphs. The basic appearance of the graphs was decided upon according to Twingly's wishes and the consideration of the best usability customs. It is important to keep the graphs as big and easily read as possible, without having to compromise on the other elements of the page. Since Rickshaw did not have all the features needed, as described in section 2.3.4, some had to be added to the library. For one, Rickshaw does not support legends connected to more than one graph. This feature was needed because some pages will contain multiple graphs. Having just as many legends as graphs is confusing for the user and steals a lot of space. Still, the graphs need some individual controls. Each graph has an individual slider for controlling the time span. This slider is used to delimitate and expand the time span of the graph. The available data spans from a year back and forward, so there has to be a way for the user to focus on a time span of their choice. At first the jQuery UI slider was used for this task and worked great in development. However, it was discovered that it does not support touch, which was a requirement for the dashboard to work on a tablet. The jQRangeSlider, as seen in figure 4.1, supports this. The end and start points for the time span are visible and the span can be inverted and slid as a whole. Default values are used for the slider handles to set the default time span of the graph to an appropriate value. A minimum is set for the slider range since a range of, for example, two days will not give the user any additional information.

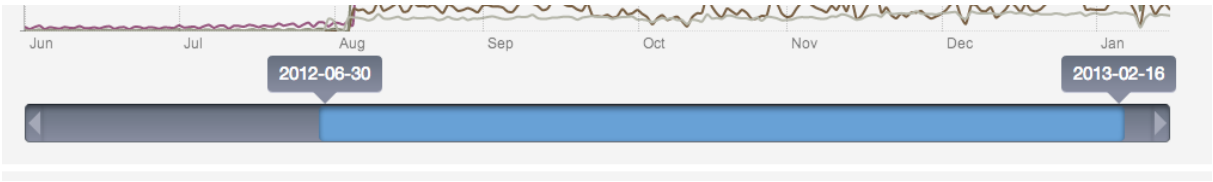

Figure 4.1: The time slider used to interact with the graphs.

A smoothing function can be enabled for every graph. Smoothing means interpolating, or softening, the curves to make it easier to interpret. A curve can be irregular which is great for detail, but also a lot for the brain to process if the user is only interested in the overall trends of the curves. The smoothing can be turned on or off by the user with a preset value. The smoothing functionality is integrated in Rickshaw. The graph lines have colors based on a palette in Rickshaw. Both the legend elements and the lines they are connected to have the same colors. This is to make it into an entity and to show that they are connected, see figure 4.2. This is also discussed in the theory in section 2.3.3. The same colors can also be found next the the number statistics on the page. This is to show which legend-bars selection/deselection affects which numbers. Figure 4.3 shows this. The total statistic number has no colors since it is not affected by the legend. The other statistic number has the colors of the legend, i.e. it is affected by selection/deselection of these.

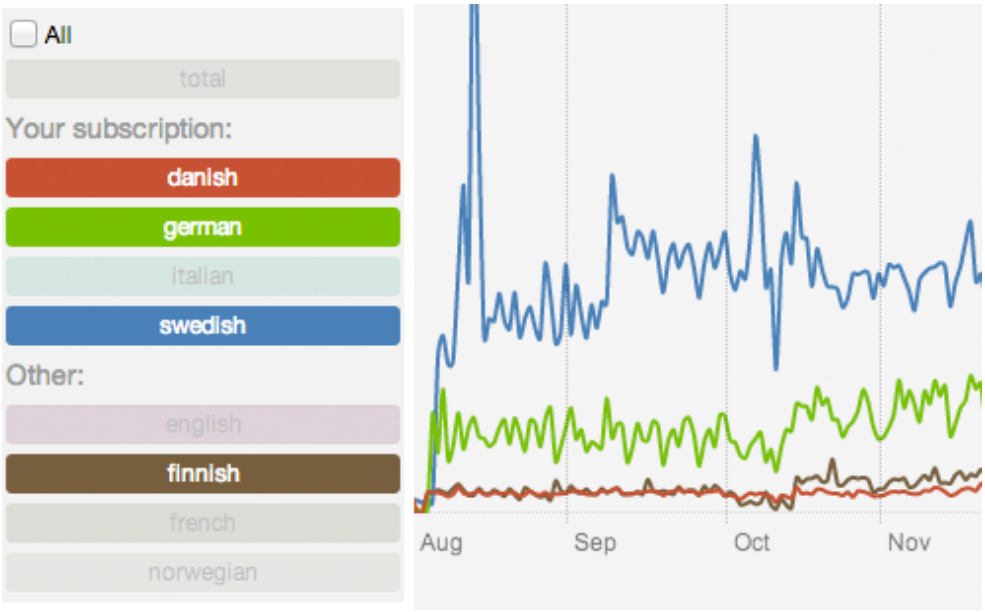

Figure 4.2: The same colors used in the graph and the legend. Left: Legend. Right: part of graph.

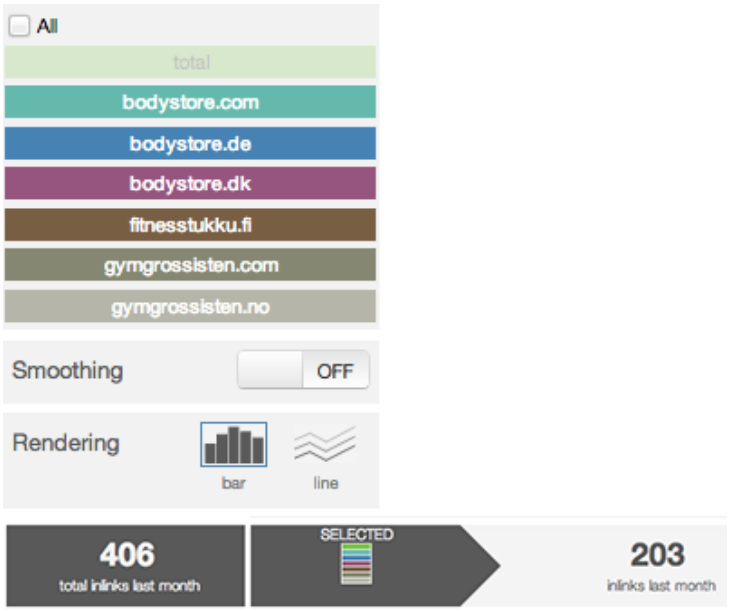

Figure 4.3: The same colors used in the legend as in the color bars next to the statistic numbers. Top: Legend. Bottom: Examples of statistic numbers.

The default legend in Rickshaw is basic, see figure 4.4. The legend was edited to make it more user friendly and fit in better with the design. A checkbox to check/uncheck all the options was added to make it more easy to use. Using a checkbox for users to control the choice between two distinguishable states is a method recommended in the rules in section 2.3.5. Using a checkbox for this is intuitive and often used, which will help the user recognize the action. Instead of using checkboxes for each label in the legend, the whole legend is used to select/deselect an option. This will also make it easier to use when the dashboard is used on a tablet. Twingly hard-coded the maximum number of options available in the legend. The 8 largest alternatives are selected and only these will be streamed to the dashboard. The default behavior in Rickshaws is to deselect an element when the checkbox is deselected and deselect all other element except the clicked element when the label is clicked. It was thought that this could be confusing for the user, since producing the desired result might not be self explanatory, as discussed in rule 7 in the golden rules in section 2.3.3. The new way, with only one way to select/deselect, makes it more clear and simple. It is more consistent, the actions are easily reversed, and increases the feeling of control since the user can be certain what will be the result of her actions.

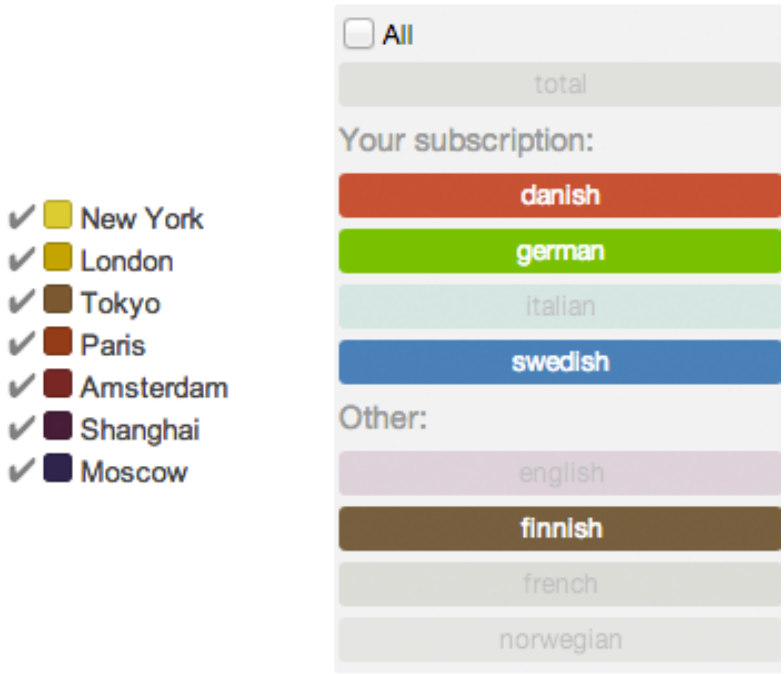

Figure 4.4: Left: The default legend in Rickshaw. Right: The new legend.

In the current version of Rickshaw, it is not possible to change the time zones. Since most of the data is in UTC/GMT +1 hour or UTC/GMT +2 hour this has to be fixed. By re-calculating and re-displaying the parts of the graphs that contain time, this problem is solved. The hover label for the graphs is one of these situations and when editing its layout, some simplifications of the date are also done. This simplification involves the date format and layout, all to minimize the space needed, but at the same time does not compromise its usability.

4.3 Different views

The dashboard has two types of views, data and widget. There is one view for each type of customer, as discussed in section 1.2. After signing in, the user is redirected and allowed access to one of these. If the user is both a data and widget customer, the user can choose which view to load.

4.3.1 Data

The data view of the dashboard has one main page, see figure 5.4 in chapter 5. This page contains three graphs and four statistic numbers. The fact that it contains more than one graph creates some problems. Each graph needs a separate legend to control it. Switching between these and keeping track of the different states gives further problems. The statistic numbers displayed on the page are intuitive to the user. These statistics update to correspond to the selected alternatives in the legend. This functionality required the dashboard to keep track of each language's value and knowing which are selected and which are not. This is done using HTML5 web store, localStorage [4]. It allows for data to be stored locally within the user's browser. Such a simple thing as setting the statistics to zero when all legend-bars are deselected is complex. Since the dashboard is constantly updating and receiving new data, each new state has to be checked at many places. This is done by stopping the updates when the different statistics are set to zero. Also, the existing animation of the numbers has to be stopped, otherwise it might keep counting even if it has been set to zero. In the legend, only the top languages are displayed to limit the number of alternatives. But in the statistic number "total new blog posts indexed today", all indexed languages are included. This creates a problem when calculating the "selected new blog posts indexed today" number. If all languages are selected, it will still not be the same value as the total value displayed. This might confuse the user, which is why it was decided to be clear with both the description of the statistics numbers, as well as the color bar next to the numbers.

The legend is well thought through to give it good usability. The standard appearance of a legend in Rickshaw is the same as in many other examples; a list with checkboxes. In this development, since the goal is for the dashboard to be modern and also work on a tablet, something different is needed. The small checkboxes can often be difficult to operate on a tablet. After many discussions and trial and error, a solution was found. Instead of checkboxes, each choice in the dashboards legends are labels/buttons. This gives a clean feeling and makes it simple to use on a smaller screen, such as a tablet. When an option is deselected the labels colors are muted and the text color changed. After doing some usability testing, it was found that a clear distinction between selected and deselected options is important. After discussions with Twingly, further changes in the legends were done. Since the data view of the dashboard will be used by customers that buy specific parts of the data, it is important to make it clear what they are and are not paying for. Both to be able to compare your data with the other data, but also to show the customers what they are missing by not buying certain data, see this in figure 4.4.

4.3.2 Widget

The widget view will be used by the customers who have Twingly's widget on their web page.

4.3.2.1 Statistics

On the widget page, see figure 5.5, some of the same changes are made to the legend. The labels are used in the same way instead of checkboxes, for the same reasons. The difference here is that the user can only see

their own data, thus making the division between "Yours/Others" unnecessary. The statistic numbers have the same layout as the data page, but they represent different things. The fact that this page is similar to the data page is important from a usability point of view. If the same user uses both pages, he/she should be able to recognize the layout and functionality.

4.3.2.2 Blog Overview

The blog overview page of the widget view does not contain any graphs, see figure 4.5. This page shows the flow of the latest blog posts for that specific user that is signed in. The purpose of this page is for the user to be able to see what people are saying about them and their products. Each post section has information about the blogger and the blog post. The blog post information is quite basic; the name of the post, the name of the blogger, blog post category, sentiment, date and time, location and type of post (inlink or mention). Inlinks are blog posts that contain links to the signed in customer and mentions are blog posts where the customer is only mentioned in plain text. For now, inlinks are in focus. The blog overview also shows the number of times that each blog post has been linked to by other blogs. Each blog section in the blog overview also has the feature that it can show an extract from the post text content. However, this feature seemed a bit meaningless since the user can navigate directly to the post and read it. This led to that feature being dropped. Sentiment is the classification if the blog post is negative, neutral or positive about the customer. At Twingly they call this the Twingly rank. The information about the blogger is mostly meta data. The gender and age group are present, as well as the category of the blog.

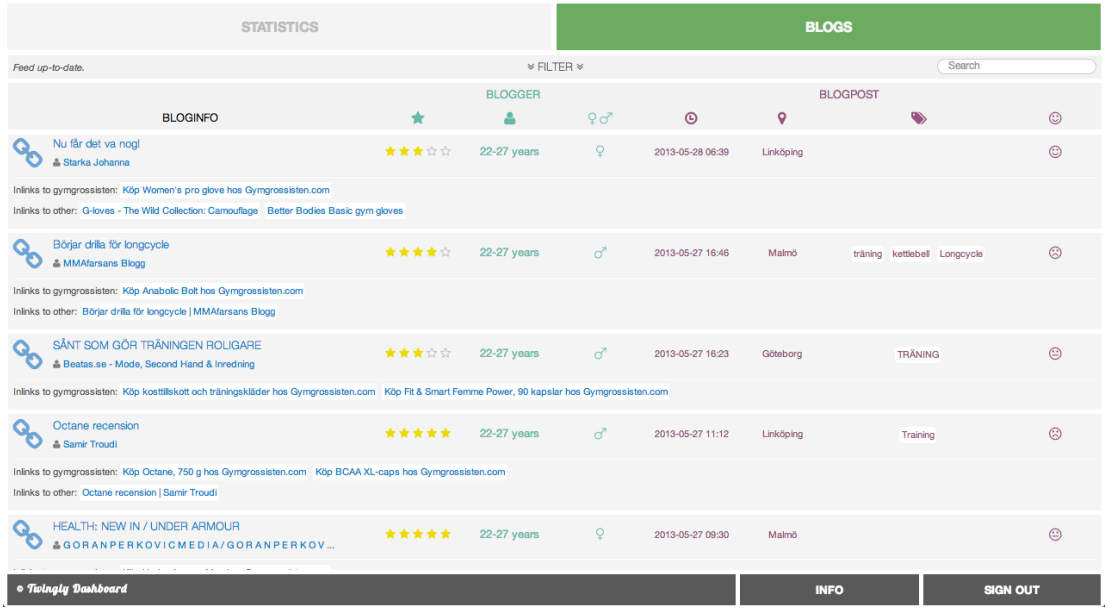

Figure 4.5: The blog overview page in the widget view.

The first thought was to make the blog overview page into a constant flow of incoming blog posts. However, this would cause some issues for the user. Imagine the user scrolling through the page and finding a post of interest. The user pauses and starts reading. Then a new post is displayed at the top of the page and the

whole page is pushed down one step. This causes the user to lose their place. This will cause the user a lot of irritation, since the user needs to feel in control, as discussed in the theory in section 2.3.3. This problem was solved by notifying that there is a new blog post and retrieving it on demand through an Ajax call. When a new post is waiting, an alert text that says how many new blog posts are available is displayed, see figure 4.6. When the user wants to see new blog posts, he/she can just press this alert text. The new blog posts are then displayed and the alert text disappears until new blog posts are available.

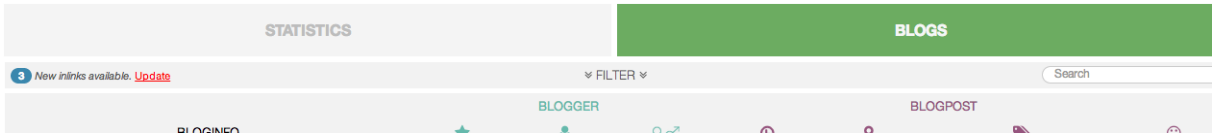

Figure 4.6: The blog overview page in the widget view when a new blog post is available.

The meta data present for most of the blog posts is also used to let the user filter the data, as suggested in section 2.3.5. This filter is partially done and will be extended in the future. The possibility for the user to filter the incoming data is important. This is in line with much of what the theory says about both information visualization and usability principles. The fact that the data can be scaled and rearranged are important rules in information visualization, as discussed in section 2.2.5. Another important aspect is to make the user interact with the data, due to the fact that interaction will enhance the dashboards effectiveness. The filter bar is designed to be intuitive, see section 5. The icons used to represent the different meta data are commonly used and narrative. They are also the same ones used to represent the meta data in each blog post, to give the user a sense of recognition and coherence, both important for good usability as said in sections 2.3.1 and 2.3.3. The icons are created by Glyphicons, provided by Twitter Bootstrap, see section 2.4.2.2. Additional icons are provided by Font Awesome, made for Twitter Bootstrap.

The user will at some point want to search the blog posts for whatever reason. A search field was created in the left corner of the filter. This listens to the user typing and initiates an Ajax search to Twingly Search. The search takes a moment to complete. While waiting to receive data, the blog posts are removed and a progress bar is displayed. When the data is received, the blog posts matching that search slides down. When designing the blog overview page many different layouts are tested. Many different types of data has to be displayed, but each blog post has to be limited in size to be able to minimize scrolling. At the same time the design still has to be effortless and clutter-free. At the far left of each post, an icon is displayed to show which type of content (inlink or mention) is provided. Each blog post has a header bar that contains the blog post title, name, date, time, location and the links. The meta data is also listed as a table. This is to make it clear to the user what is what. It is better to be excessively clear than to leave anything to chance, this page might get a bit confusing to the user if it is not structured. The table structure also makes it easier for the user to compare the meta data of the different blog posts. At the bottom the links are listed. The links are blue, see figure 5.7, as this is a commonly used color for links. There are many different ways to get this data. In the beginning of the development, the data is found by getting an id from the AMQP stream and querying the database for the blog post data. Later, the data is retrieved using an API that makes it simpler to select the right data for the right customer, likewise minimizing the dashboards direct interaction with Twingly's databases.

4.4 GUI

As previously explained, Twitter Bootstrap is used in the dashboard since it offers a standardized element design and typography. This speeds up the process of creating elements inside the application. Black and white are used as base colors, as recommended by Badre in section 2.3.1. Twingly's green color is then used as an accent color. Also, blue is used for links, since this is a commonly used color for links. This gives the user a sense of recognition as discussed in 2.3.3. Keeping views simple is a key goal. Simplicity is an important aspect covered in 2.3.3 rule 8. As discussed in section 2.3.1, one of the key aspects of usability in web development is to make the user avoid thinking about the user interface. It should be obvious and self-explanatory. This is something that is constantly in mind when developing the dashboard. First impressions, discussed in section 2.3.1, is important on a web page. It is in the developers interest to keep the look and feel of the dashboard as modern and simple as possible. One aspect of this is to design the dashboard as a sort of billboard. All parts of the page are always present, even if they are not showing. The navigation between the different pages is animated with a slide motion, to further fortify the feel of moving over the billboard. This is all in line with what the theory says in section 2.3.5 and section 2.3.3. It is important to give the user a feeling of recognition and familiarity. The "billboard feel" is also a good choice since it should be possible to use the dashboard on a tablet. The big buttons and sliding behavior of the dashboard gives it an "app" feel when used on a tablet. In section 2.3.3 of the theory it is explained in rule 3 that it is important to have informative feedback. One occation when this is essential for the dashboard is upon login. An incorrect username or password gives feedback to the user, so the user knows what went wrong and can fix the problem.

4.5 Users

The first page the user encounters before logging in contains an explanation of what the dashboard is. Even though the dashboard should be self explanatory, this is thought to give the user some tips and tricks to get the most out of the dashboard right from the beginning. Each customer has its own user login, present in Twingly's databases. The data customers do not have a login yet, but this is to be added to Twingly's databases. When the user signs in, there are a few things that could happen. If the user is a data customer, the data view is loaded. In the same way, if the customer is a widget user, the widget view is loaded. In some cases, the user is both a widget- and data customer. If this is the case, the user has to select which view to be redirected to. Having signed in, the user information is stored in a session cookie. The information stored is the user name, user type and a session id. The user is only allowed access to the view of its user type.

4.6 Testing

Testing is performed on a regular basis during the development and small bugs are often fixed right away, while larger problems might be made into its own task. Functionality tests are done during the programming, both by the developers and the mentors at Twingly. Performance testing is done after each finished task.

4.6.1 Usability testing

Usability tests are carried out on real users in a real environment. However, usability testing on real users is simply not possible prior to having a good enough product to show the customers. The usability test are therefore carried out in the second half of the project. Usability tests are also be carried out on people who are not the "real" users. This to get a feel for the general usability of the dashboard. Since the intended end user can be anything from a novice to an expert, all types of usability testing are of value. Every week the progress is presented to the mentors at Twingly. This meeting, in addition to being a status meeting, is also used as a sort of usability testing. They get to navigate through the dashboard and try out new features. This makes it easy to see what features need better usability. Things that are obvious to the developers can be difficult to understand for the average user. Such small things as the color of buttons and element margins sometimes make a big difference to the usability!

4.6.2 Performance testing

During development, the performance of the dashboard is tested. Performance means what the user experiences. One test is to leave the dashboard running for several hours, to make sure that it does not time out or crash. Another thing is to stream a lot of data to the dashboard to see if it can handle large amounts of data. The maximum number of users connected at one time might be a problem, but since the dashboard will only be available to a limited number of users, it will not be a problem at this point.

4.6.3 Functionality testing

Functionality testing is done consistently during development. Each finished task has its functionality tested and evaluated. In addition, the developers at Twingly also tested the dashboard functionality during the development.

5 | Results

All this work resulted in a dashboard with a clean and user friendly design. Many of the goals set up in the beginning, see section 1.1, were met and others have changed during the process of development. The focus was to have a finished and stable product with all the basic features by the end of this project, so that it could be launched. In the beginning, see section 1.1, of this project several goals were set up. One of these was to be able to have real time updates in the dashboard. This has been achieved by using AMQP, see section 2.4.1.1, to set up queues. Another goal was to let the user interact with the data in several ways. This is done in all of the different views. All graphs can be altered using the time sliders to set a time interval. They are also altered by selecting or deselecting alternatives in the legends and by turning the smoothing on or off. The statistic numbers present on two of the views are also affected by the legend. The blog overview on the widget page, section 5.7, displays the recent blog posts containing links to the customer from a month back. A filter can be made visible by pressing the filter bar. The filter does not effected the posts. It will have to be implemented later. A user can search for posts containing keywords, whilst still linking to the customer.

5.1 Start

The start page, seen in figure 5.1, contains a short explanation of the dashboard. It is not neccessary for the user to read this prior to using the dashboard for the first time, but it might give a headstart.

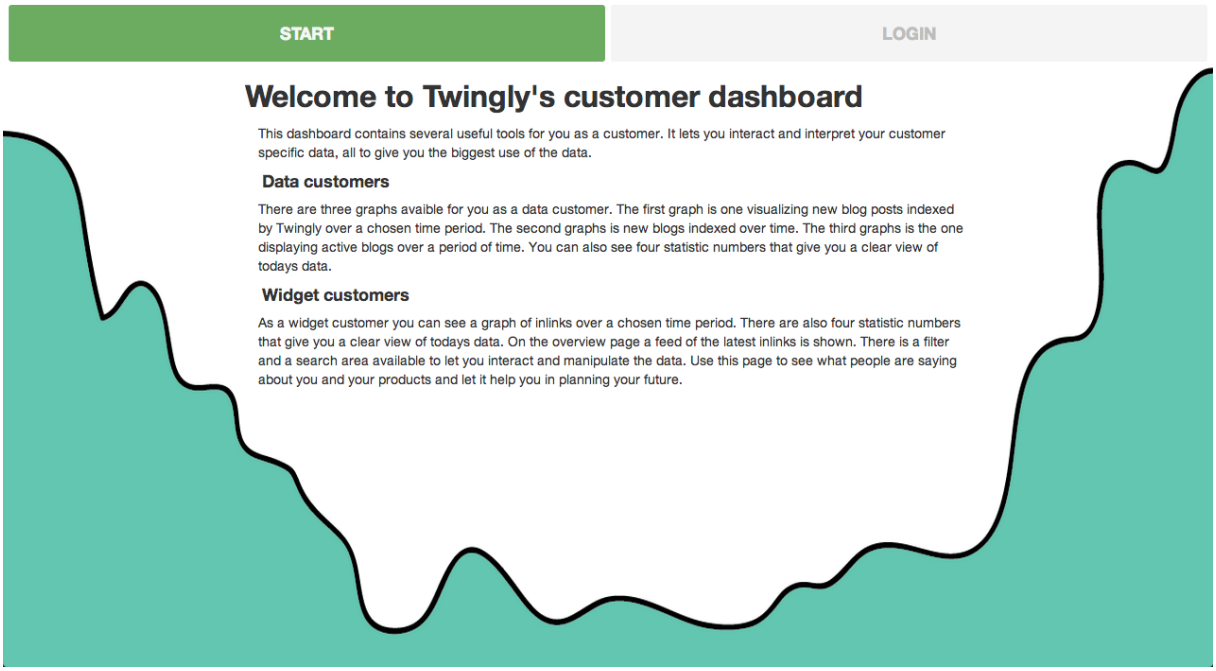

Figure 5.1: The start page with instructions.

The page seen in figure 5.2 shows a screen shot of the login page. The login page is clear and simple, which will make it usable on a tablet as well. Users need to log in to see the visualized data, both to protect Twingly's data from unwanted users and to be able to do customer specific visualizations. This was one of the main goals at the beginning of this development.

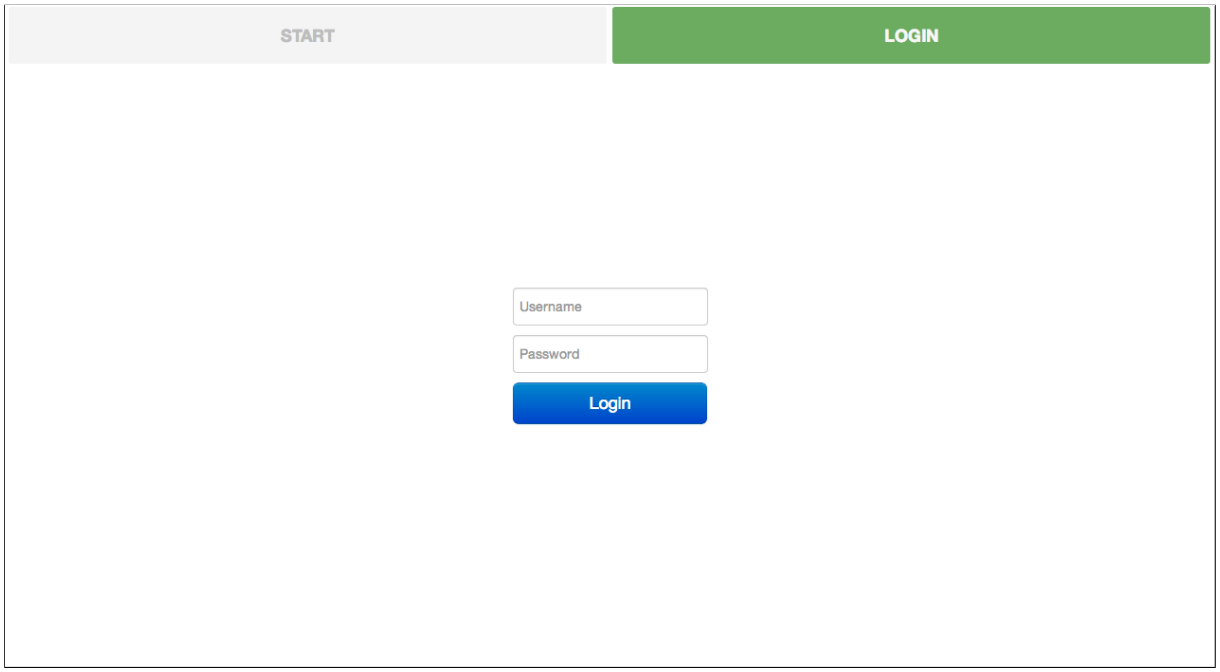

Figure 5.2: The login page.

When the user fails to log in, either by entering the wrong username or the wrong password, an error message is displayed, see figure 5.3

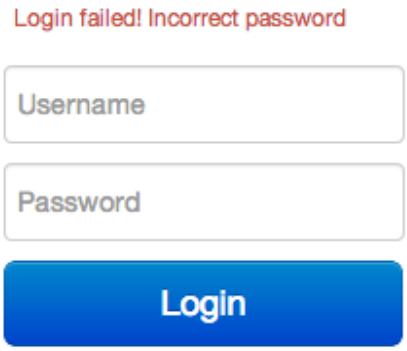

Figure 5.3: Failed to log in.

5.2 Graphs

Since the data stretches over a year the user needs a way to explore smaller time frames. For this, a slider is used to delimit and expand the time span of the graph. There has to be a way for the user to focus on a time span of their choice. The jQRangeSlider, as seen in figure 4.1, supports use on a tablet and also share a similar aesthetics as Twitter Bootstrap components. The end and start points for the time span are visible and the span can be inverted and slid as a whole. Default values are used for the slider handles to set the default time span of the graph to an appropriate value. A minimum is set for the slider range since a range of, for example, two days will not give the user any additional information. The lines in the graphs have colors from the standard color palette in Rickshaw. The colors were set so that the first labels in the legend are the most colorful. This because these are the ones that are of most imporant to the user. Both the legend elements and the lines they are connected to have the same colors to make it seem like an entity and to show that they are connected, see figure 4.2. The same colors can also be found next the the statistics numbers on the page. The colors will symbolize which legend items the numbers are based on and when a legend color is deselected it will deselect in this color bar symbol as well. Figure 4.3 shows this. However, the top element in the legend is the total and does not affect the displayed numbers. Therefore, it is not included in the color bar symbol. The legend is well thought through to give it good usability. Instead of checkboxes, each choice on the dashboard legend is a label/button. This provides a clean feeling and makes it simple to use on a smaller screen such as a tablet. When an option is deselected the labels colors are muted and the text color changed. A checkbox was added to the legend to check/uncheck all the options to make it more easy to use.

5.3 Data

The data view of the dashboard has one main page, as seen in figure 5.4. This page contains three graphs and four statistic numbers. The graphs are all loaded at the same time, but only one is visible. The user can toggle between the three by clicking the label buttons. This action will switch the viewed graph and the legend will slide up and a new one will slide down. This is a clear indication to the user that the data has changed. The statistic numbers displayed on the page are intuitive to the user. They give a direct and clear indication of todays data. These statistics update to correspond to the selected alternatives in the legend and when new data is received. When they change, the numbers visually increase to the current number. To make it obvious that the numbers are linked to what is selected in the legend, a symbol mimicking the legend is placed on the left side of the numbers.

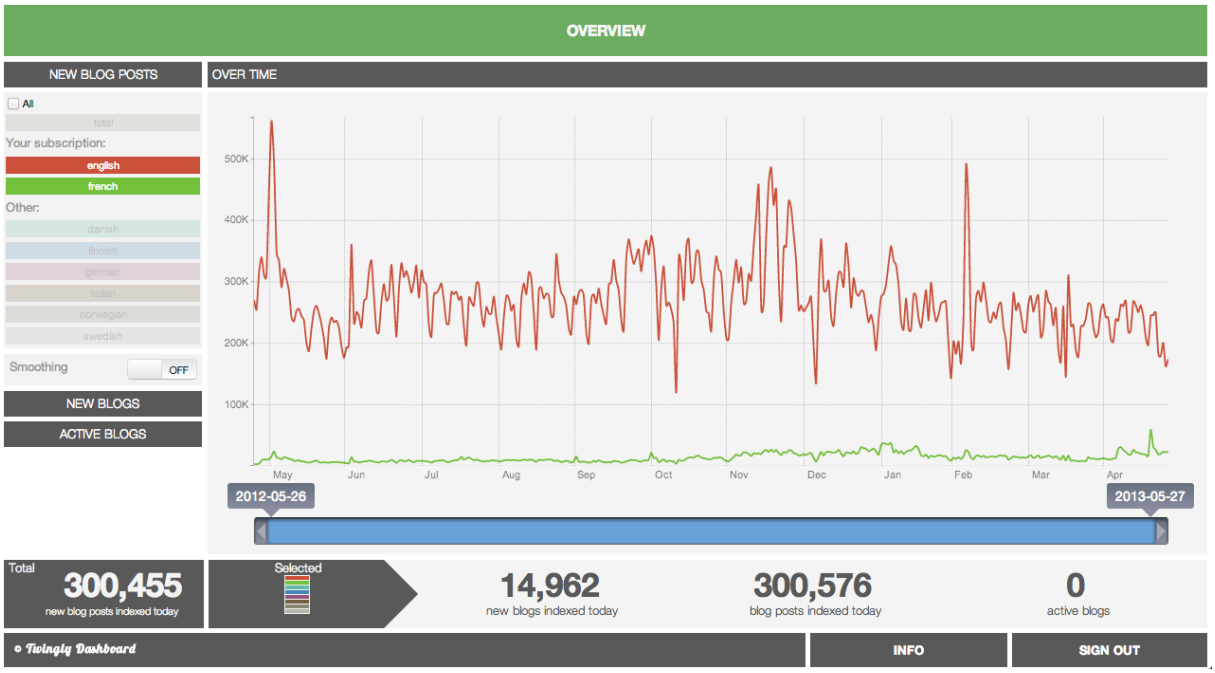

Figure 5.4: The overview page in the data view.

5.4 Widget

The widget view has the same simple design as the data view, but with some extra features.

5.4.1 Statistics

On the widget page, some of the same changes have been made to the legend. The labels are used in the same way instead of checkboxes, for the same reasons. The difference here is that the user can only see their own data, thus making the division between "Yours/Others" unnecessary. See the layout in figure 5.5. It was of great importance to keep this page similar to the corresponding page in the data view. This so that users who are both data and widget users can recognize the layout and functionality.

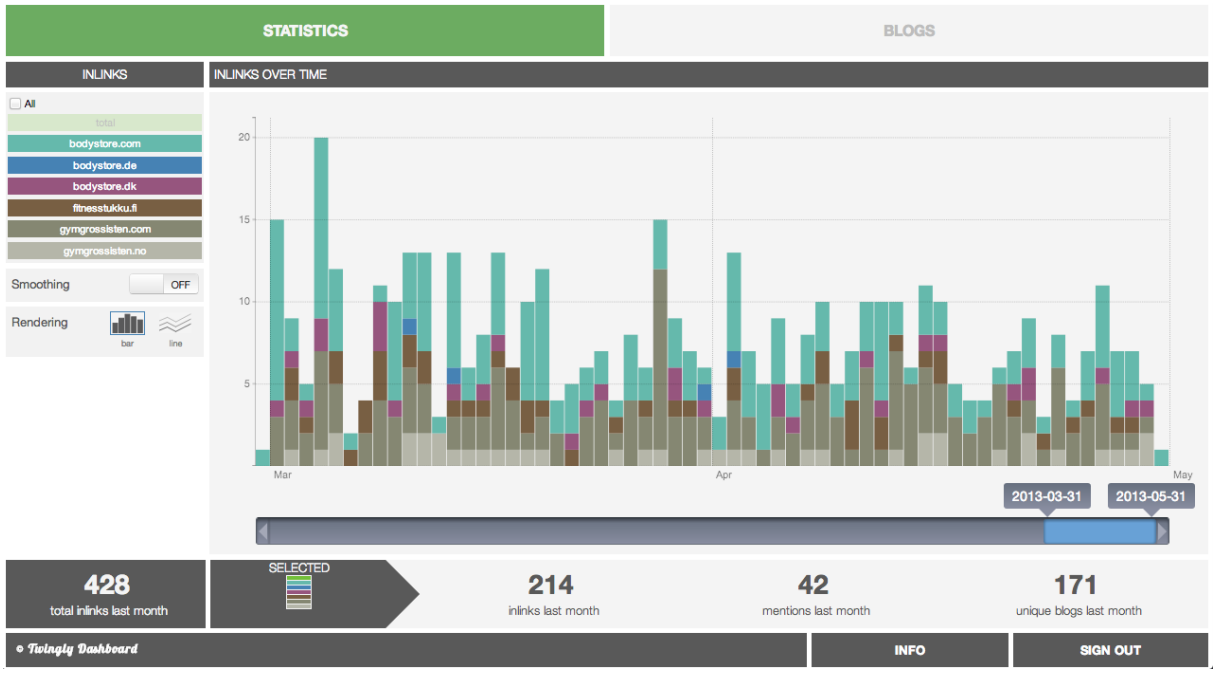

Figure 5.5: The statistics page in the widget view.

One difference is that the user can choose between displaying a bar chart and a line chart, where the bar chart is default. This selection is possible because the data values in this graph often ranges between 0-10 inlinks per day. These low values might therefore be easier to display in a bar chart, since a line chart becomes ragged, see figure 5.6.

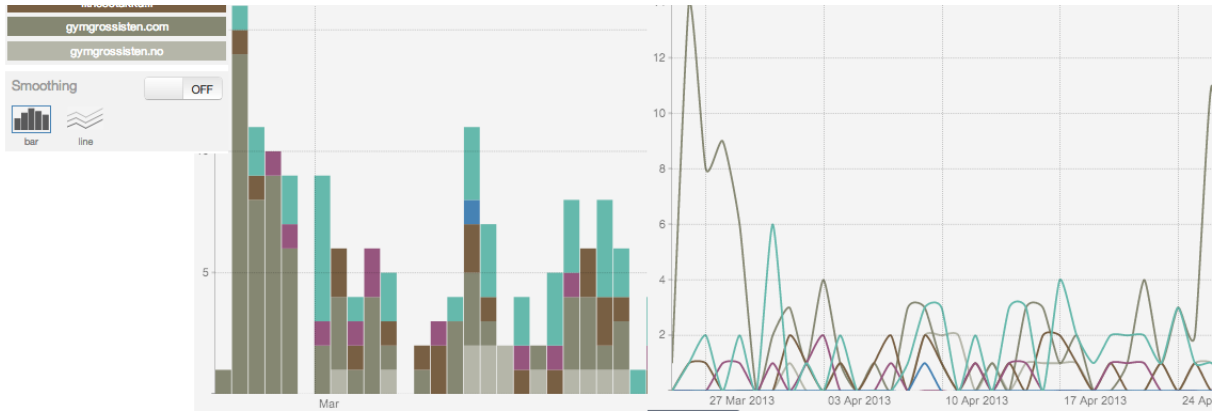

Figure 5.6: The graph as a bar and line chart.

5.4.2 Overview

The blog overview page of the widget view does not contain any graphs, see figure 4.5. This page shows the flow of new blog posts for the specific user. The purpose of this page is for the user to be able to see what people are saying about them and their products. Each blog post has information about the the blogger and the blog post. The blog post information is quite basic; the name of the post, the name of the blogger, blog post category, sentiment, date and time, location and type of blog post (inlink or mention). It also shows the links, i.e. the inlinks, at the bottom.

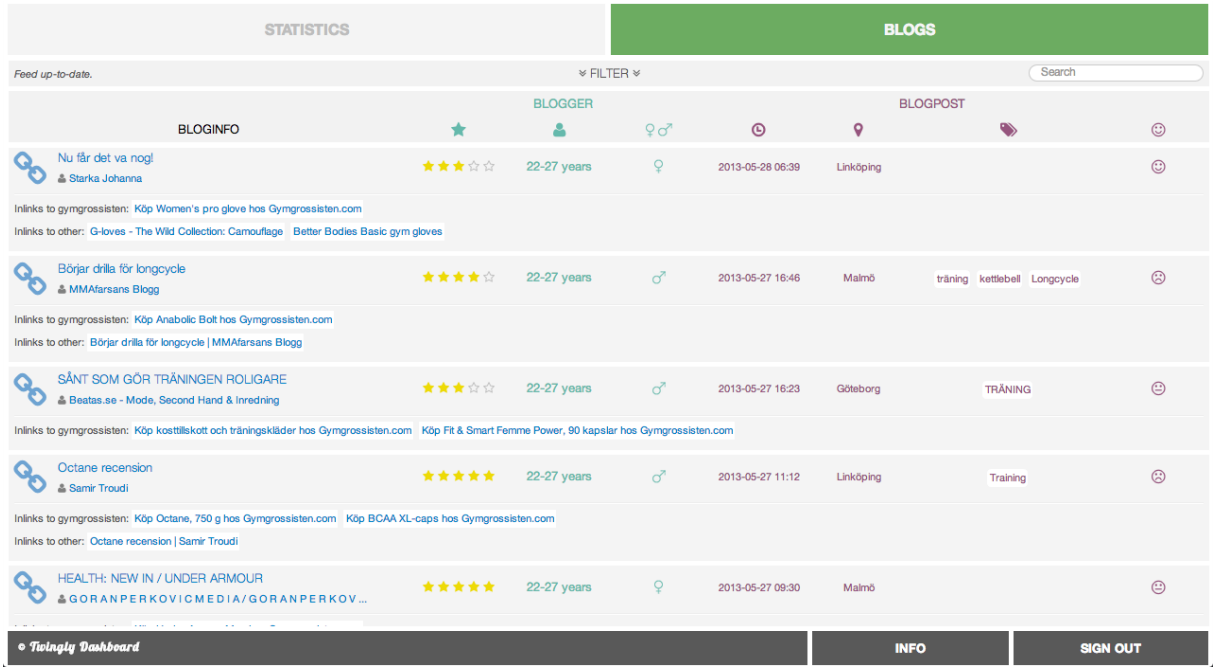

Figure 5.7: The blog overview page in the widget view.

At the far left of each blog post, an icon is displayed to show which type of content (inlink or mention) it is. Each blog post contains the blog post title, the blog title and the meta data. The links are shown at the bottom. An alert text that says how many new blog posts are available is then displayed, see figure 5.8. When the user wants to see new blog posts, he/she can just press this alert text. The new blog posts fetched by an Ajax call and then displayed. Afterwards, the alert text disappears until new blog posts are available.

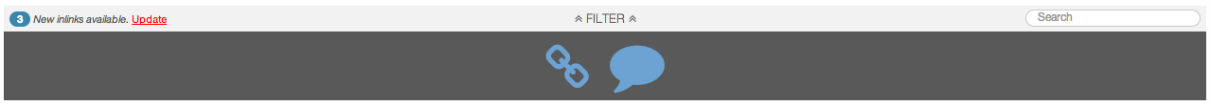

Figure 5.8: The filter and search field on the blog overview page. At the far left a counter for new blog posts can be seen.

The filter bar is designed to be intuitive, see figure 5.8. The icons used to represent the different metadata are common and narrative. They are also the same ones used to represent the metadata in each blog post. At the moment the search which loads the blog content only retrieves the posts that have links to the customer (inlinks) and not posts that only mention the customer without linking (mentions). This means that the filter bar is purely visual as of now. The thought is to add more filter options in the future as the correct metadata becomes available. The icons are created by Glyphicons, provided by Twitter Bootstrap, see section 2.4.2.2. Additional icons are provided Font Awesome, made for Twitter Bootstrap. In the upper right of the page is a search field, see figure 5.8. This field is used to search the users data, as discussed in the theory in 2.3.5. The placement of the search field in the upper right corner was due to the fact that this is where it is often located on websites. This gives the user a sense of familiarity.

6 | Discussion

The work was carried out in an agile fashion and was dynamic. In the process of making the prototype for the dashboard features were added to the project backlog. Whenever a task was completed a new task was pulled from the backlog. Both pair and individual programming was utilized. The weekly status meetings with Twingly have been a large part of the agile workflow. The meeting gave us a chance to perform usability testing, at the same time as new ideas and suggestions were brought up. Twingly often came with clear instructions of what they wanted and we took these and made them our own. We came with further suggestions and this led to good discussions. The layout and looks of the dashboard do not follow a traditional website standard. At the same time, a dashboard is not a traditional, everyday website. The GUI that evolved during the development was inspired by the user experience design of other dashboards, such as Captain Dash[1], and the simplicity of smartphone apps since the dashboard is more like an "app" than a website. Something that has slowed down the development has been the fact that most of the data was not readily available at the beginning of the project. Since the data was so scattered throughout Twingly databases and servers, it often took a lot of work to get the simplest type of data. It often meant that help and guidance was needed to find the data. Sometimes, the developers at Twingly did not even know where the data was to be found. This often slowed down the development, as this Master's Thesis Project was not always prioritized. If the data had been available from the start, or at least most of it, the development would have come a lot further. The fact that data was found during the development also meant that finished parts had to be redone several times, since the input data had changed once again. This was sometimes frustrating as it led to a lot of extra and unnecessary work.

When it came to the choice of languages and frameworks there is one major thing that could be questioned looking back. We decided to use Rickshaw instead of using d3.js directly, as we thought we would be using basic functionality. In retrospect it might have been better to use d3.js since we ended up doing a lot of changes to the basic Rickshaw framework. If we would have used d3.js we could have done the same things but built it from smaller parts, instead of taking the finished graph elements and editing the parts that we wanted to change. Other than this, the other languages and frameworks used in the project worked out well. Sinatra and Ruby were both simple and fun to use. jQuery was also perfect for this project and saved us a lot of unnecessary code. We have learnt a lot of new techniques during the project and feel like it really developed our programming skills. There were some pages in the dashboard that were developed but later discarded. This was due to several factors. One cause was the fact that the page's functionality was added to other pages. Another cause was that the data needed for the page was just not available at this moment. Since the dashboard is to be released to customers we, in agreement with Twingly, choose to exclude these pages from this version of the dashboard. The fact that some of the pages had to be discarded was of course not what we wanted in the beginning. Putting time and effort into things that will not be used in the end product is never fun. It was a case of "kill your darlings" as these parts did nothing for our project at the

moment, so it was best to exclude them. If there was time, one wish was for the dashboard toork on tablets, changing layout according to screen size, as well as on regular computer screens. Th the moment it works well in landscape view, but the layout is not responsive to the screen size. Therefore, portrait view is not optimal for the dashboard. Something that could have been done in a different way is the testing. We did basic testing and some usability testing with everyday people. It was the plan to test with a few of Twingly's customers earlier in the development, but for this to be done the dashboard had to have real data. Since it took until the end of the project to get real data, the opportunity to test was first presented in the last weeks of the development. At the end, usability testing with a customer never happened. The development dragged on and it got to close to the end of the development to have time for usability testing. At the end of development we byfound ainatra-based framework for creating dashboards hathat d been released. This framework is called Dashing[21]. This seems to be a great framework and would have been an interesting approach if we had known about it when we first started. It has been great working at Twingly. The mentors at Twingly were always close by for answering questions. The fact that the dashboard will be launched and actually used by Twingly's customers is something that is really fun! When we begun the development, we did not yet know of how much importance the dashboard would be to Twingly's future business strategy. The dashboard will play a big role in Twingly's international launch. It will also be a lot of fun to see what the dashboard will become in the future.

7 | Conclusion

The end result is a dashboard with good usability and a clean look, as desired in section 1.1. Twingly are satisfied with the result and will use it as a first version to launch to their customers. The graphs used to visualize the data are effective, with interaction that most users can understand, as discussed in section 2.3.1. The graphs are easy to use and give the user a lot of information. The legends used for the graphs are customized to work well on both a computer and a tablet. The time slider used for the graphs are also compatible with a tablet. The blog overview page gives the user a lot of information about new blog posts, what people are saying about them and what type of people are visiting their site. The use of the dashboard requires a login and this enables the dashboard to be customer-specific. The data is continuously streamed to the dashboard, making it work in real-time. The dashboard is stable and every part of it has been carefully considered and thought through. The goals were reached and the end result is close to what was first planned at the beginning of the project. Twingly will launch it to the intended users in the near future as a part of their launch into the international market.

8 | Future work

At the moment, Twingly plans to launch this new application as a vital selling point for expanding internationally. It will be launched as a first version as it is today. More functionality and features are already planned to be added and Twingly will expand the dashboard in the future. The pages excluded from the development are still to be developed in future releases. Today, Twingly does not store all data that is needed to create some of the wanted features. Our version of the dashboard contains all the basic features. In the near future the Twingly team will add to it with update releases. The dashboard works great on both the desktop computer and tablets such as iPad, but there are other platforms and screen sizes out there. After having released the first version of the dashboard there should be an evaluation of how the users are viewing and using it. The way in which the data is streamed to the graphs will probably be developed further in the future, to get a faster dashboard.

Bibliography

[1] http://captaindash.com/. Accessed: 2013-05-17.

[2] http://ghusse.github.com/jqrangeslider/. Accessed: 2013-03-04.

[3] http://sequel.rubyforge.org/. Accessed: 2013-04-02.

[4] http://www.html5rocks.com/en/features/storage. Accessed: 2013-05-16.

[5] Badre Albert. *Shaping web usability*. Addison-Wesley, 2002.

[6] Schneiderman Ben Card K. Stuart, Mackinlay D. Jock. *Readings in Information Visualization using vision to think*. Morgan Kaufmann Publishers, 1999.

[7] Rachel Andrew Ben Schwarz Lea Verou David Storey Christian Heilmann Dmitry Fadeyev Marc Edwards Aarron Walter Aral Balkan Stephen Hay Andy Clarke Elliot Jay Stocks, Paul Boag. *The Smashing Book 3*. Smashing Media GmbH, 2012.

[8] The PostgreSQL Global Development Group. http://www.postgresql.org/. Accessed: 2013-03-04.

[9] Haase Konstantin Harris Alan. *Sinatra: Up and runnning*. O'Reilly Media, 1st edition edition, 2012.

[10] Shutterstock Images. http://code.shutterstock.com/rickshaw. Accessed: 2013-02-22.

[11] National Cancer Institute. http://www.cancer.gov/global/digitalguide. Accessed: 2013-06-03.

[12] The jQuery Foundation. http://jquery.com/. Accessed: 2013-03-04.

[13] SIMON HERBERT A. LARKIN JILL H. *Why a Diagram is (Sometimes) Worth Ten Thousand Words*. Carnegie-Mellon University, 1987.

[14] Cournoyer Marc-André. http://code.macournoyer.com/thin/. Accessed: 2013-03-04.

[15] Bostock Michael. http://d3js.org/. Accessed: 2013-02-22.

[16] OASIS. http://www.amqp.org/. Accessed: 2013-03-04.

[17] Thornton Jacob Otto Mark. http://twitter.github.com/bootstrap/. Accessed: 2013-02-22.

[18] Hipp D. Richard. http://www.sqlite.org/. Accessed: 2013-02-22.

[19] Spence Robert. *Information visualization*. AMC press books, 2001.

[20] Plaisant Catherine Shneiderman Ben. *Designing the user interface.* Pearson Education, 4th edition edition, 2005.

[21] Shopify. http://shopify.github.io/dashing/. Accessed: 2013-05-30.

[22] Krug Steve. *Don't make me think. A common sense approach to web usability.* New riders publishing, 2006.

[23] The Haml Team. http://haml.info/. Accessed: 2013-02-22.

[24] Edward R. Tufte. *The Visual Display of Quantitive Information.* Graphic Press, 2nd edition edition, 2001.

[25] Hetherington Victoria. http://www.dashboardinsight.com/articles/digital-dashboards/fundamentals/the-dashboard-demystified.aspx. Accessed: 2013-05-17.

[26] Denise Jacobs Christian Holst Jamie Appleseed Colleen Jones Iris Ljesnjani Vitaly Friedman, Elliot Jay Stocks. *The Smashing Book 3 13.* Smashing Media GmbH, 2012.

[27] W3C. http://www.w3.org/html/. Accessed: 2013-03-04.

[28] W3C. http://www.w3.org/style/css/. Accessed: 2013-03-04.

[29] Matsumoto Yukihiro. http://www.ruby-lang.org/en/. Accessed: 2013-02-22.

Appendix
A | Prototypes

Figure A.1: Data customer - Overview page.

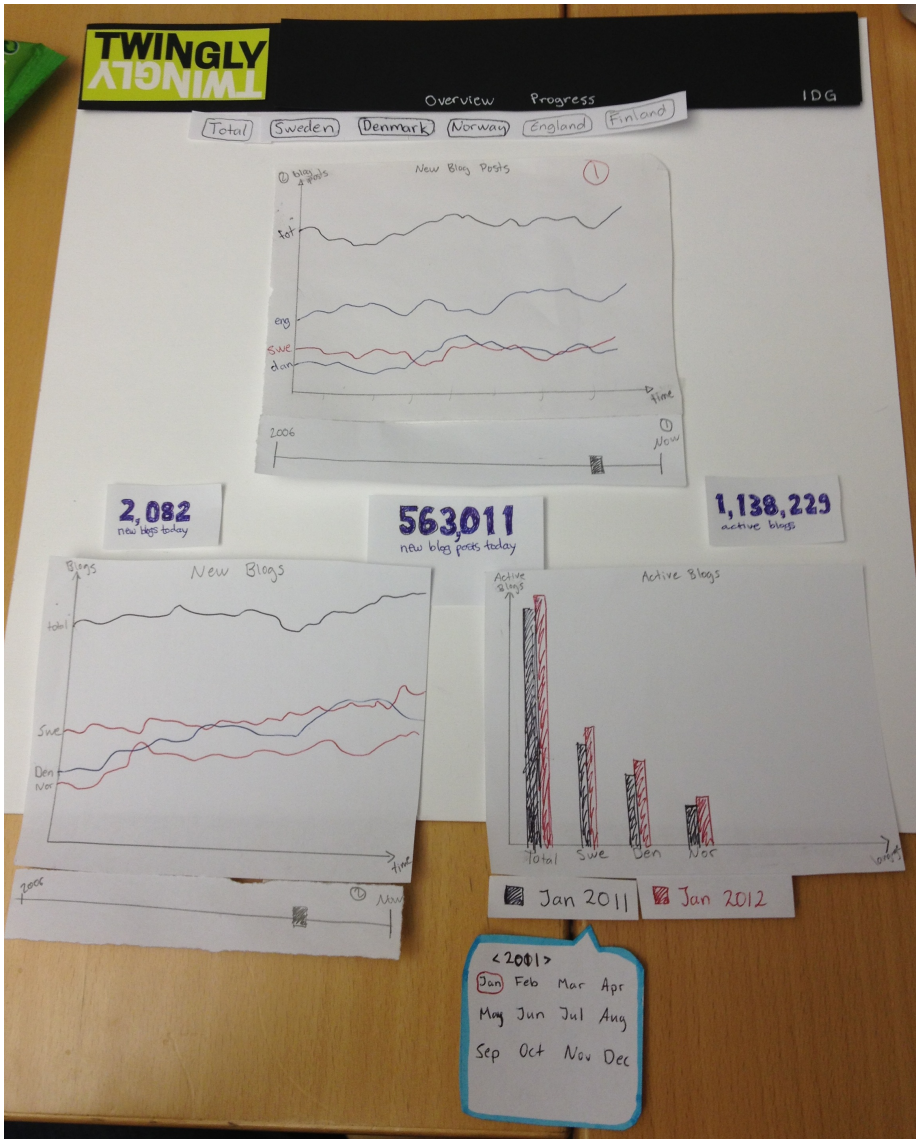

Figure A.2: Data customer - Overview page.

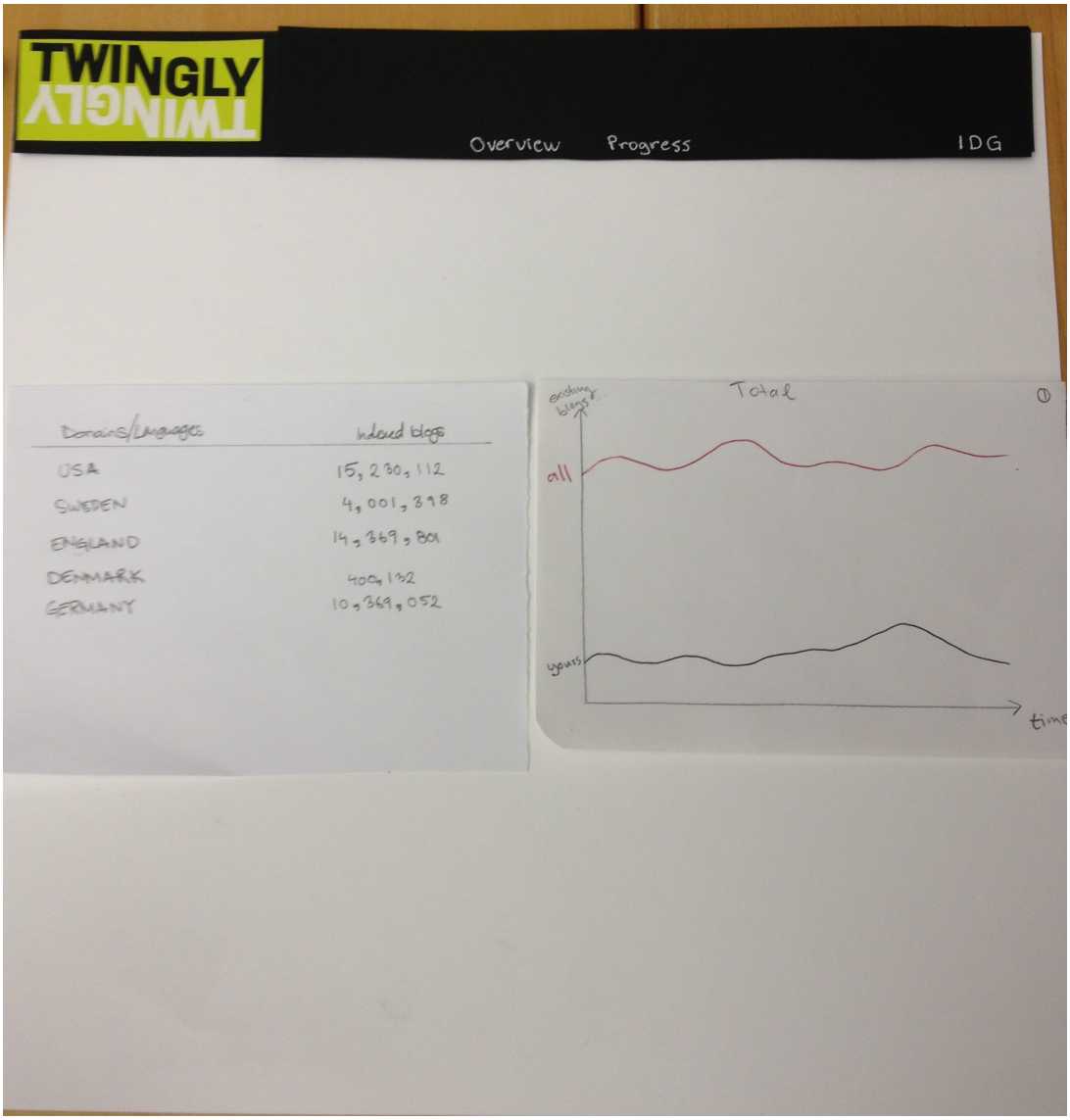

Figure A.3: Data customer - Progress view.

Figure A.4: Widget customer - Ability to choose domain.

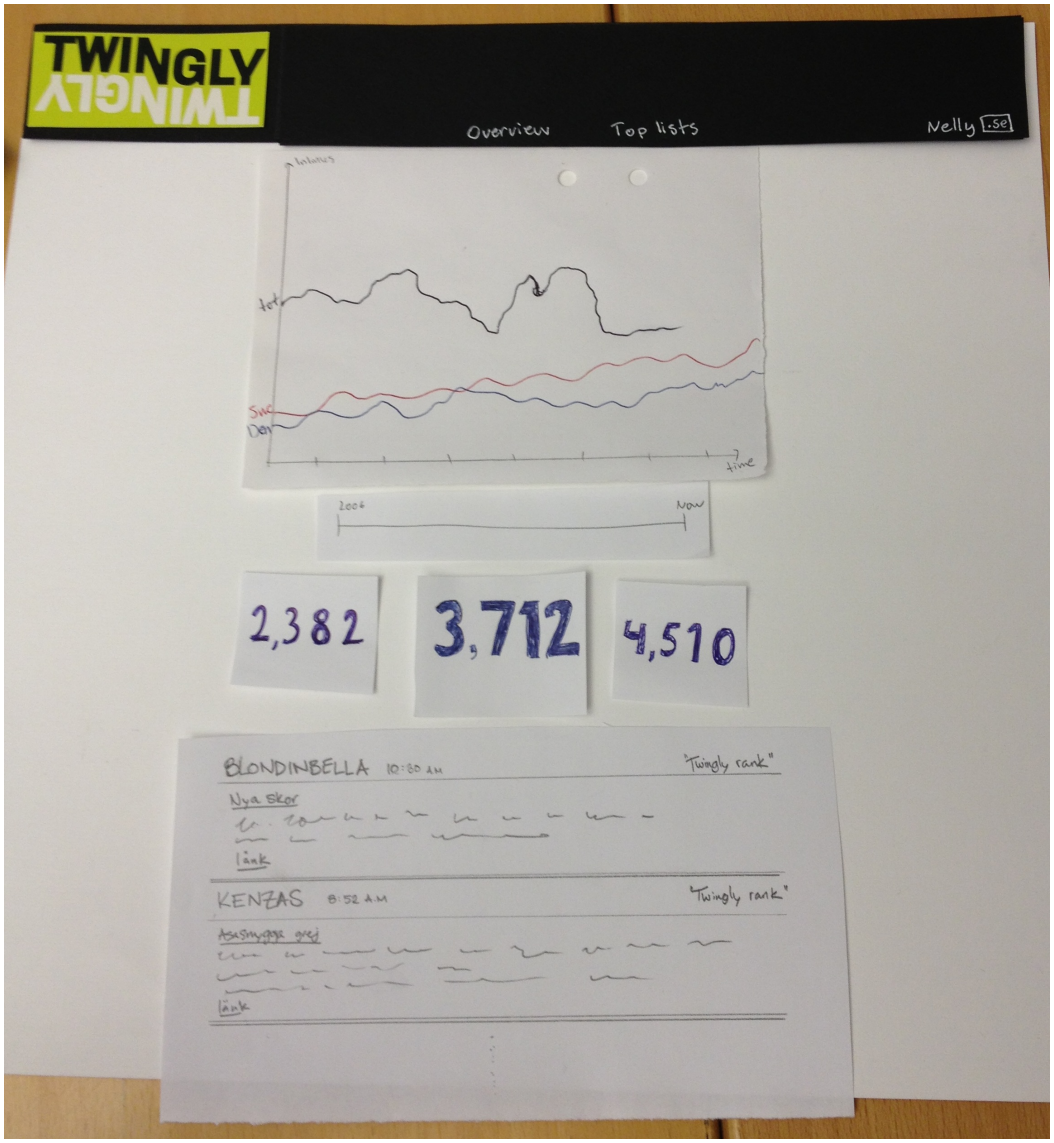

Figure A.5: Widget customer - Blog overview page.

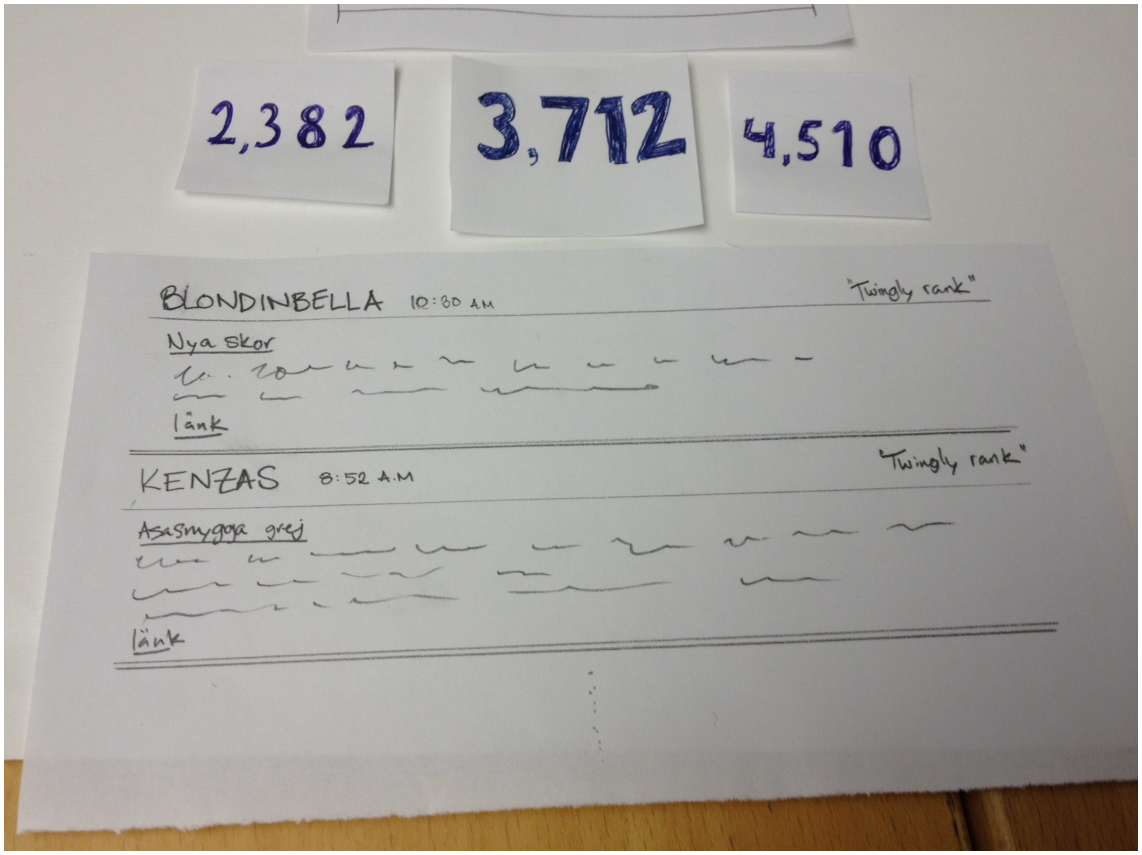

Figure A.6: Widget customer - Latest blog posts.

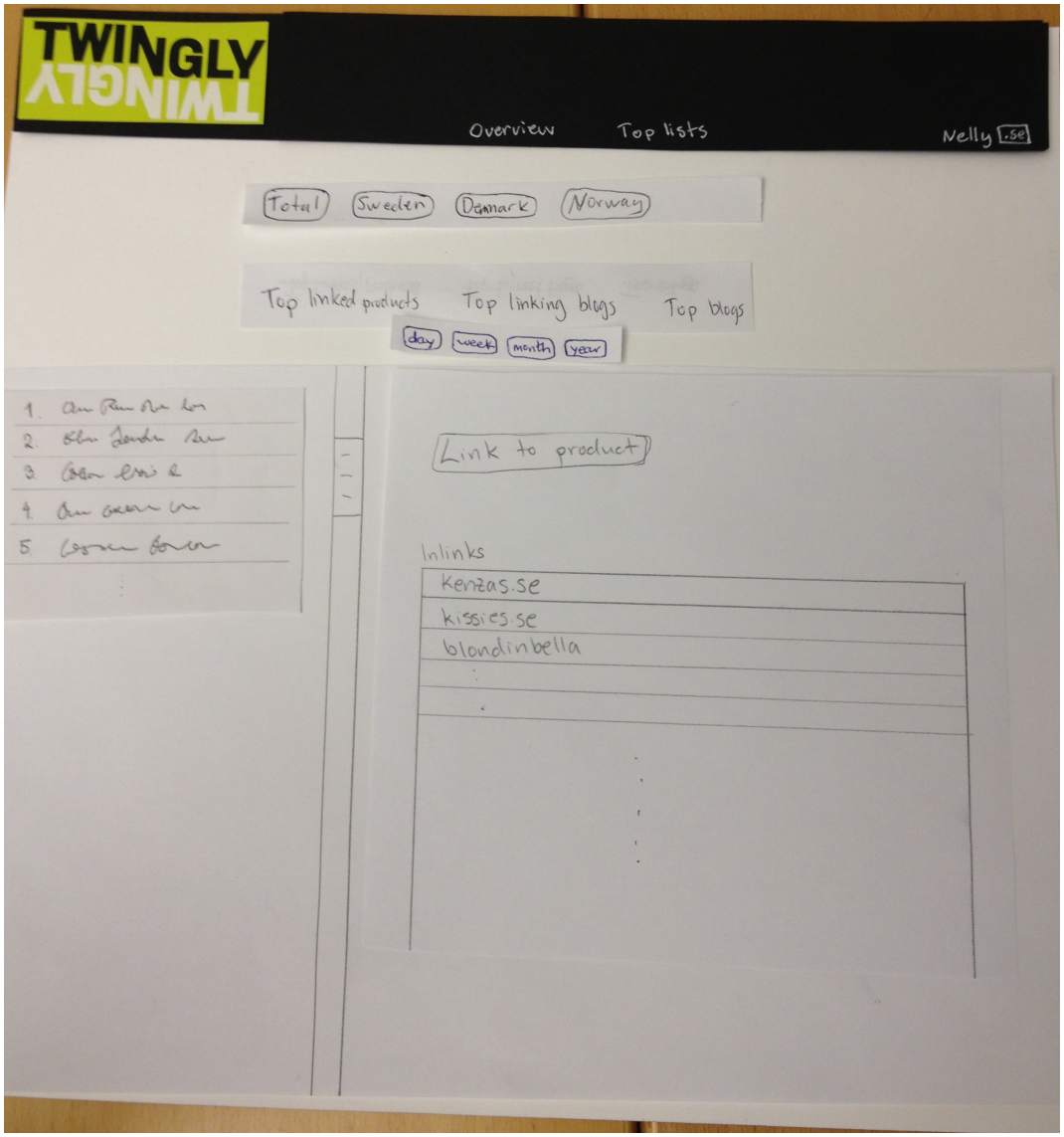

Figure A.7: Widget customer - Toplist view.

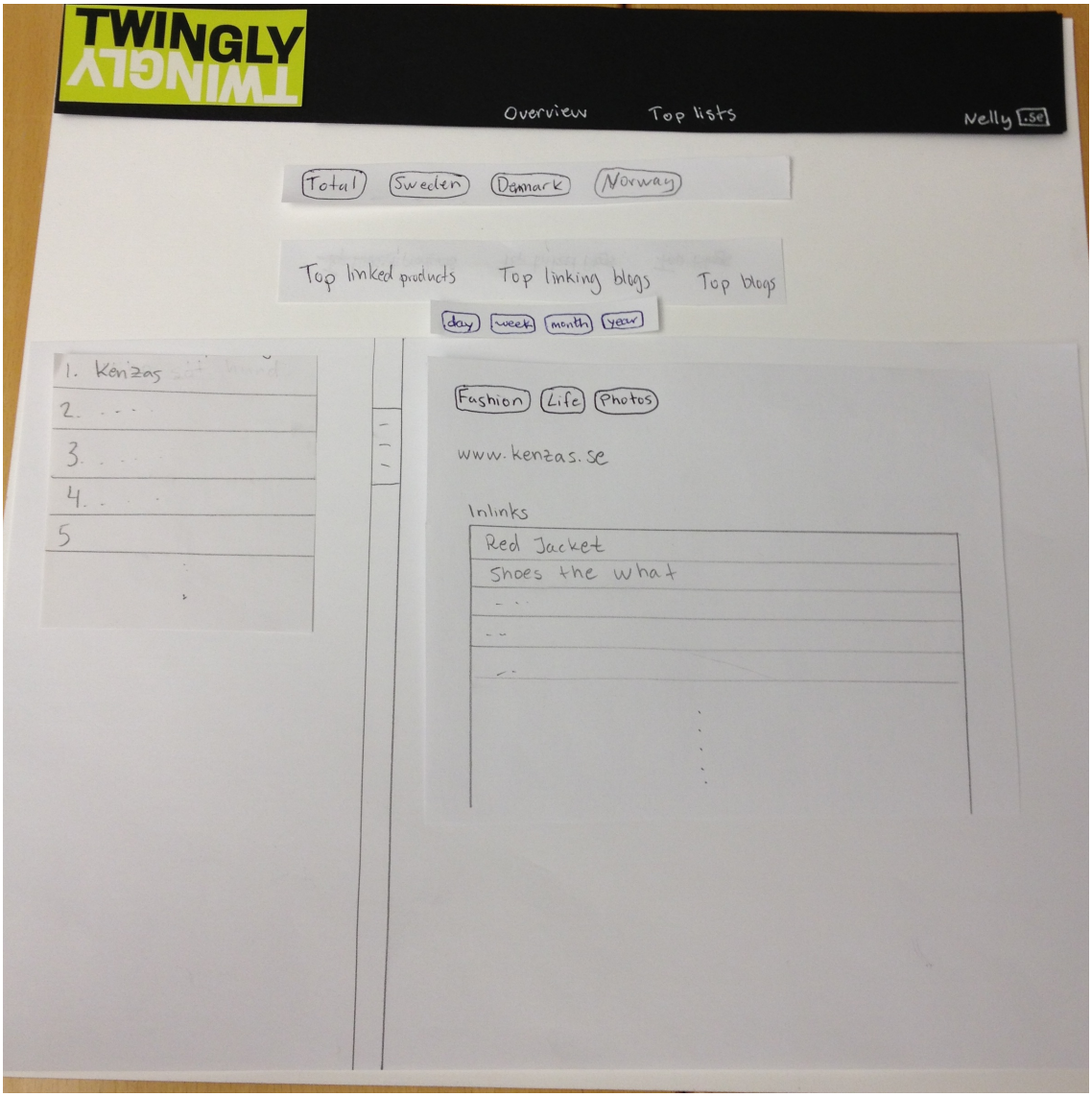

Figure A.8: Widget customer - Toplist view.

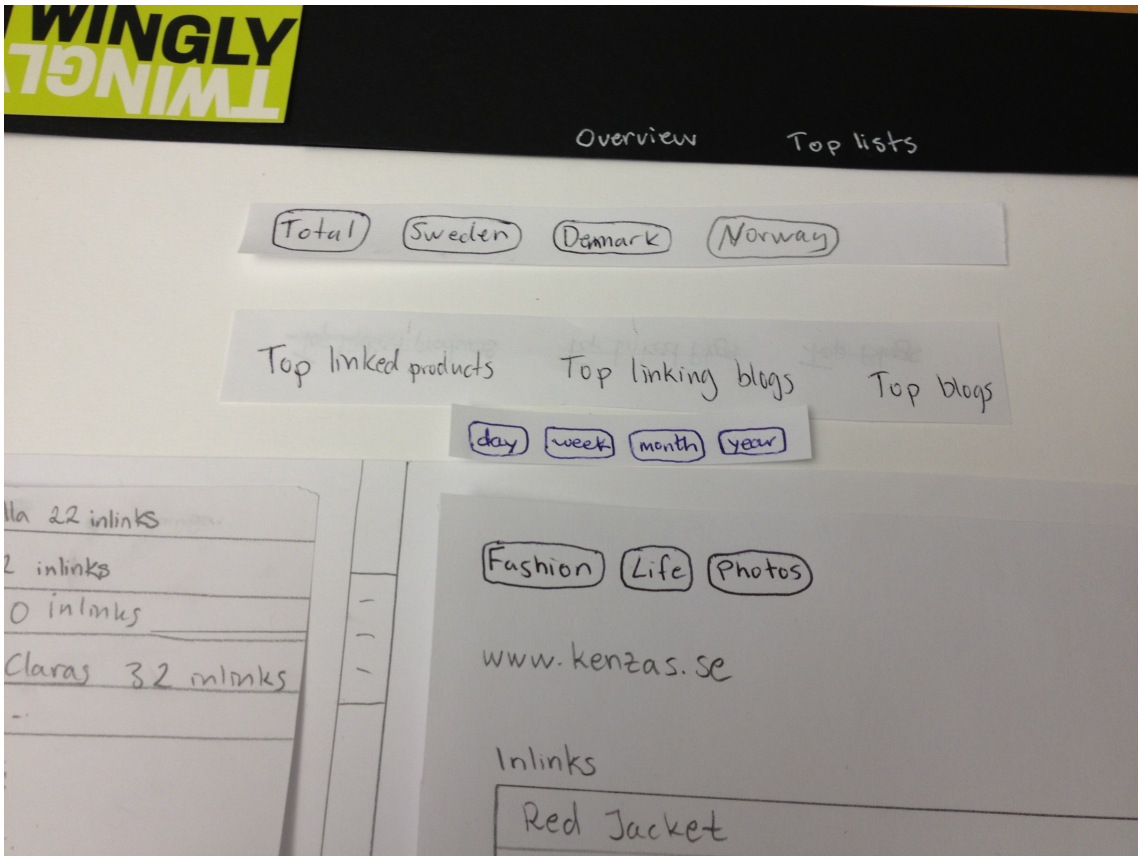

Figure A.9: Widget customer - Toplist view close up.