Ram Joshi
Walter Koning
Chulki Lee

Masters Final Project Report
School of Information
University of California, Berkeley

May 4, 2012

Advisor: Coye Cheshire

Table of Contents

Executive Summary

What is Grepvine

Grepvine brings great discussions to your browsing experience. A handy sidebar shows comments and articles related to the web page currently open in your browser. Filters enable you to discover richer user comments and articles. Grepvine currently supports technology news on Engadget, Mashable and Wired. We are adding support for more sites very soon.

Grepvine is incredibly easy to use. Just install the Chrome extension. Then browse your favorite technology news. Grepvine will be there for you. Grepvine only shows up when it has useful information related to the current web page and hides away on other pages.

Grepvine has been designed from the ground up to be quick and friendly.

Grepvine was inspired by the success of news aggregation sites such as Digg, Slashdot and Reddit in encouraging rich discussion threads. Grepvine works like a giant collection of discussion threads promoting the most relevant conversations on diverse topics across different websites and comment platforms.

Grepvine has something for every reader. Richer comments can be discovered through filters such as popularity and word count. Users can also explore the most relevant or the most commented upon articles related to the web page you are reading.

Etymology of the name

grep·vine (n).

1. A Chrome extension that brings great discussions to a browser near you.
2. a. The informal transmission of information, gossip, or rumor from person to person.
2. b. A usually unrevealed source of confidential information.

Root word from Unix grep.

grep [options] PATTERN [FILE...]
grep searches the named input FILEs (or standard input if no files are named, or the file name - is given) for lines containing a match to the given PATTERN. By default, grep prints the matching lines.

Usage

"I read it through the Grepvine" - anon.

Origins and Justifications

The Status Quo

The focus of our project is the online discussions space. Our goal with Grepvine is to promote online discussions by addressing problems with the current design and implementations of online comment platforms.

Online comment platforms in the form of forums, or message boards, have been around since before the internet as we know it (world wide web) became popular. In the 1970s to mid-1990s BSSes (Bulletin Board Systems) were a popular form of online communication. Today, comments on web pages are an important forum of impersonal communication and discussion. Public comments can help users gauge opinion about current topics.

We believe that a free flow of online conversations is what makes the internet fun, useful and the best medium of expression that we have. However, increasingly the web is being divided into silos. Social networks favor communication with friends over impersonal communication. Search engines and websites increasingly customize contents to users' historical interests. Comment systems on public websites do not allow users on one site to communicate with users on other sites talking about the same topics.

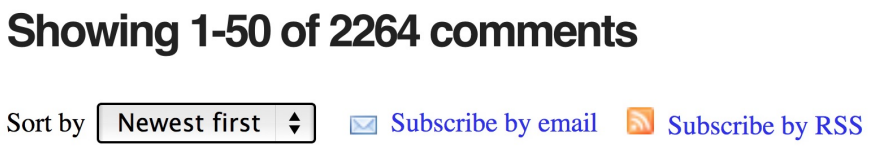


Figure 1. There are thousands of comments on popular articles

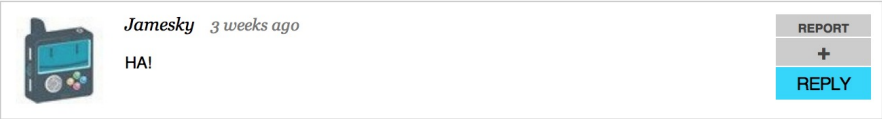


Figure 2. Many comments are short and irrelevant for good discussion

Comment threads on popular articles tend to get very noisy (Figure 1. and Figure 2.). News

aggregation sites such as Digg, Reddit and Slashdot allow users to promote comments and articles by voting on them. Useful information is promoted through democratic upvoting (or liking) and noisy information is demoted through downvoting (or disliking). Content gets moderated and ranked without explicit manual interference. We studied the salient features of these sites and tried to incorporate some of them in the design of Grepvine.

The Problem

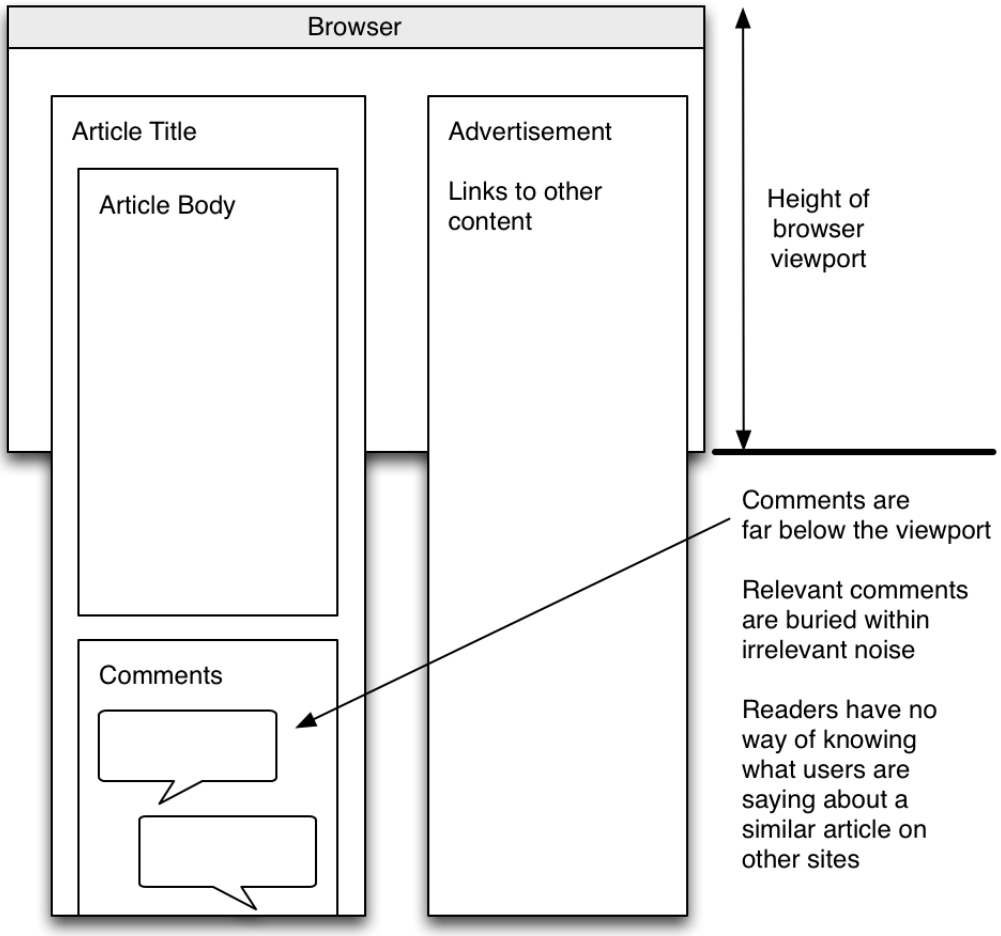


Figure 3. A diagram showing how comments are normally placed below the article.

Current site layouts neglect comments

Most designs of blogs and news sites display comments below the main article body. This causes the comment form and user comments to be pushed far below the fold of the viewable window area, the browser viewport. This makes comments both harder to access and less valuable as a source of information.

Information Overload

There is too much duplication of content, especially on popular current events in technology and

politics.

No ideal way to find online discussion

There is no exclusive search for online comments and opinions. Search engines treat comments as part of the text on web pages. There is no curation.

Search bubble

Contents on websites are carefully customized to user preferences and there is a growing concern of a search bubble. We believe that a free flow of online discussions is one of the ways to escape this trend. Current tools limit free exploration.

The Solution

Overview

At the heart of our solution is the task of determining document similarity. Our solution works by first looking at the content that the user is reading in the current browser tab. This information is sent to a server where similar documents (articles) are found. Using the collection of similar articles we determine a set comments about the the article that the user is reading. The best matching similar articles and comments (determined by document similarity and user filter criteria) are then sent back to the user and displayed in a sidebar inside the current browser tab. The entire process of finding similar articles, related comments, filtering, ranking and rendering results occurs online (at runtime).

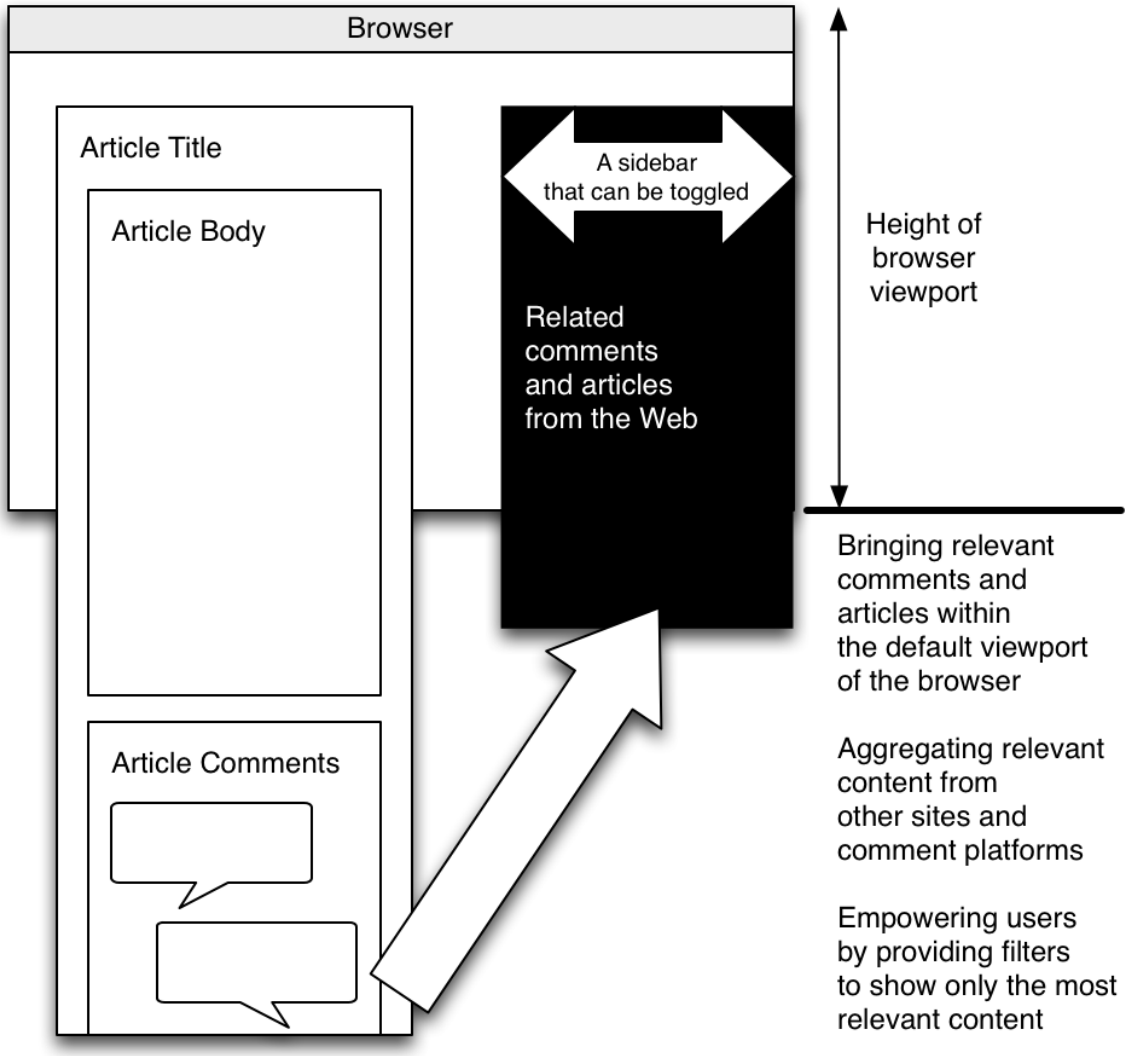


Figure 4. The Grepvine Browser Extension appears as a sidebar overlay that can be "pulled" from the side of the browser window.

Document store

Our solution requires a large corpus of documents for finding related content. The current implementation of the solution is using a corpus of documents from three sources: articles from Mashable, Engadget and Wired. Newly published articles are scraped every hour. The solution can be scaled very easily by adding more sources of documents to the corpus. We use MongoDB (a NoSQL document-oriented store with dynamic schema) running on an Amazon EC2 instance for storing our corpus.

Document similarity

There are several ways to find similar documents. Well-known solutions include tf-idf, n-grams, Bayesian classifiers, Jaccard's similarity, Support vector machine, crowdsourcing etc. We are using Solr, an open source enterprise search engine built on Apache Lucene project to index our corpus and search for similar documents.

Sidebar

Our solution uses a sidebar embedded inside the current web page open in a user's browser to show related comments and articles. This is achieved by using a browser extension. The current implementation of Grepvine uses a Chrome extension. The solution can be easily scaled to Firefox and other browsers by implementing extensions or plugins for those browsers.

The sidebar is positioned on the right column of the web page where it does not overlap the main content. The sidebar can be toggled in and out of view and only shows up when it has useful information to present to the user. The most important design feature is that the sidebar is fixed to the top right of the browser viewport and scrolls independently of the web page it is embedded into. This allows the reader to scroll and read the main web page independently of content inside the sidebar.

Accessible

The Grepvine solution is accessible to people of all computer skill levels. It can be installed by just one click on the Google Chrome browser is extremely simple. Users can help the computer-challenged too without worry about an ongoing commitment.

Our Product

Features

When readers arrive on an article on a website supported by Grepvine they may see pull tabs on the top right corner of their browser window. Grepvine can distinguish between article pages and non-article pages on websites based on the url pattern and the pull tabs only appear on article pages. These pull tabs have icons indicating comments, articles, and statistics related to the article being read. The number next to the related comments icon indicates the number of comments that we have identified as relevant to the article. The number next to the related articles icon indicates the number of related articles.

Product Walk Through

Install our Chrome Extension.

A user can find our extension in the chrome web store. A link to download the extension is also available at http://grepvine.info.

Browse to a Grepvine supported websites

We currently support three online publications. These include Mashable, Wired and Engadget. A user can browse to an article on any of these sites and Grepvine will display a sidebar with related content from our corpus.

Interact with Grepvine pull tabs

On the top right side of the page a user will see five icons. The first icon will open and close the viewer to see our product. The second icon displays the number of comments that Grepvine has determined to be related to the article you are currently reading. The third icon displays the number of related articles. The fourth icon is for statistics relating to the Grepvine result set. The fifth icon is a link to a page about our project.

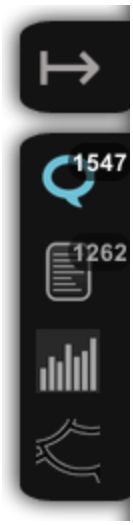


Figure 5. The “pull tabs” provide access to Grepvine content.

Interact with the location bar icon

In the browser’s location bar there is an icon that allows a user to turn the extension on and off for a specific page. This will make the sidebar including the pull tabs disappear completely for that page only.

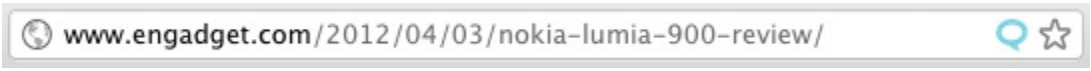


Figure 6. Notice the blue icon in the location bar.

View related comments

A user can click on the related comments pull tab to discover comments from multiple publications. Grepvine empowers users to find more interesting and relevant discussions. We do this by first analyzing related articles and then further analyzing the comments on these articles. Comments can then be filtered to include all comments from related articles, or only comments from the current article a user is reading. Comments are sortable by the following facets: popular, word count, replies, and random.

Figure 7. Screenshots of the comments tab with each sorting feature selected

Comments are displayed with an abundance of additional information. Some of this information is available immediately. The rest can be accessed through an intuitive interface.

The user will notice an avatar image for the person who posted the comment. The user name, publication, and time elapsed since the comment was posted are all standard for each result. Up to 120 characters of each comment are displayed immediately. And a user can click on the comment or the ellipsis below it to see the remainder of a longer comment.

More information related to comments may also include the word count, the number of replies to a comment, the time and date of the post, and the number of upvotes. This information is available for individual comments by clicking on the small circles above the avatar image. Alternatively, a user may click on the circles above the entire search results to view information for all comments at once.

When a user selects a sorting feature the results prominently display information related to the selection. This may include the number of upvotes, the number of replies to a comment, and the number of words in the comment.

When there are replies to a comment an icon appears below the avatar. A user can click on this to see the comment and replies as an overlay over the article they are reading. This feature is experimental and may take a moment to load so be please patient.

View related articles

A user can click on the related articles pull tab to discover articles related to the article they are currently reading. Grepvine has indexed and sorted articles based on keywords and article content. Results are sortable by relevance, comment count, recency, or by random.

Figure 8. Screenshots of the articles tab with each sorting feature selected

The sorting feature selected is reflected in the visual display of our results. We also use the same background color for the sort feature and in the results. For example, when comments are selected we display the number of comments for that article, presented over a blue background, the same color as the sort feature.

This design supports two of the ten general principles of user interface design. One principle is the "visibility of system status" heuristic. This heuristic says, "The system should always keep users informed about what is going on, through appropriate feedback within reasonable time."

The other principle is the "recognition rather than recall" heuristic. This heuristic says, "Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate."

Building Grepvine

Managing the Project

Task management and documentation: Redmine

An invaluable step we took when we began working together on the Grepvine project was to set up Redmine. Redmine http://www.redmine.org is a project management web application including a issue tracker, wiki, forums, calendar, a gantt chart and email notification. This enabled us to communicate in an efficient manner about project related activities, bugs and features. With Redmine our team remained focused on the activities we had to do, rather than the administration of the activities.

History

Updated by Chulki Lee 2 months ago #1

(By Ram)

> *i think for the first version we may not even need classifiers (bayes, svm etc.). a good approach could be just tf-idf vector based cosine-similarity search.*
>
> *we don't really have to implement this, since the lucene search engine already does this very efficiently. we can use pylucene for this http://lucene.apache.org/pylucene/.*
> *do you have any other ideas in mind?*
>
> *so we basically let lucene index our entire db of documents every night, then when we see a new url we just search against lucene's index to find the top n closes matching results.*

Updated by Chulki Lee 2 months ago #2

Sounds great - let's check the features - for instance, does it support search by document, in addition to by query keywords?

Updated by Ram Joshi 2 months ago #3

> *does it support search by document, in addition to by query keywords?*

Good point. Not sure yet, but a document is essentially a query with a very large number of keywords. Most likely we will have to build a query from our search document first.
Lucene has similar documents feature http://wiki.apache.org/lucene-java/LuceneFAQ#How_do_I_find_similar_documents.3F but that requires the query document to be in the index so I'm not sure if this can be done online (on th fly).

Another point.
Lucene runs on the jvm and pylucene is essentially lucene with a jcc wrapper around it. This may be somewhat difficult to setup and maintain compared to a pure python implementation.
A good alternative is http://pypi.python.org/pypi/Whoosh/ which looks more or less like a pure python lucene.

So, maybe it's best to try Whoosh at this point and only think of lucene if it doesn't fulfill our needs?

Updated by Ram Joshi 2 months ago #4

A few resources on implementing Whoosh.
http://andrewwilkinson.wordpress.com/2011/09/27/beating-google-with-couchdb-celery-and-whoosh-part-1/
http://www.arnebrodowski.de/blog/add-full-text-search-to-your-django-project-with-whoosh.html

More importantly.
Whoosh may not be fast enough for the amount of text we are indexing. It may be an order of magnitude slower than Solr for a large index.

Better alternatives.
1. Haystack for Django http://haystacksearch.org/ with Solr search engine http://lucene.apache.org/solr/
2. Djapian for Django http://code.google.com/p/djapian/ with python-xapian search engine http://xapian.org/docs/bindings/python/

Updated by Chulki Lee 2 months ago #5

Yeah I found that pylucene documentation is pretty poor (the site does not provide a link to repo.. WTF?)

I didn't check how solr REST API is extensible and flexible, but I think it would be better than using python-wrapper.

We need to customize ranking algorithm for articles or comments, because we want to use more info than just text. For instance, we need to rank articles based on comments on them!

So we might to

- a) put our ranking algorithm or parameters into existing search engine (solr, xapian, ...)
- b) build ranking algorithm on the top of search engine or library(lucene, Whoost)
- c) build everything from the scratch (nltk, numpy, scikit...)

I haven't use none of them, so I don't know how much complex it would be to use them. I want to pick easiest way, not the fastest, if the performance is acceptable.

Updated by Chulki Lee about 1 month ago #6

- **Status** changed from *New* to *In Progress*
- **Target version** set to *Week 7*

Updated by Chulki Lee about 1 month ago #7

- To use haystack, we should set up django application. Many haystack tasks are done in manage.py of django.
- I'll work on how to use solr with mongodb (or mysql/RDBMS, if required)

Updated by Chulki Lee 25 days ago #8

- **Project** changed from *Better Crawler* to *Better Website*

Updated by Chulki Lee 14 days ago #9

- **Status** changed from *In Progress* to *Closed*

Close it. we highly depends on Solr for searching for keywords or related documents.

Figure 9. Making product decisions and tracking changes with your Redmine.

Document and file sharing: Google Docs and Dropbox

We also made a conscious decision to use Google Docs for collaborative document work, and Dropbox for sharing files. Google Docs encouraged team members to contribute stubs of the work that had to be done. Other team members then had a document to edit, rather than having to start from scratch on their own. This supported our desire to work in an agile manner. A team member could start working when they had something to contribute without coordinating about who was going to start what and when. Using Dropbox allowed us to have a common file system for working documents. Emailing documents makes it difficult to manage the files. Dropbox made it easy. And it helped us synchronize the most current document to use.

Source version control: Git and Gitolite

For code management and version control we chose to host our own Git repository using Gitolite. Gitolite https://github.com/sitaramc/gitolite. allows users to setup git hosting on a central server, with very fine-grained access control and many more powerful features. Version control enabled us to separate coding work amongst the team members without having to worry about over-riding each other's code. I would recommend using a Git repository for anyone working together on a team coding project.

Meeting frequently

We made other decisions early on that helped us manage or project and stay on track. We agreed to meet weekly on Monday for four hours. This served as a hack session. We could collaborate and make decisions that would be more difficult on our own or via telecommunication methods. We also agreed to meet once a week with our advisor on Thursday, with an hour long team meeting beforehand. The weekly check in with Coye allowed us to show off what we had done in the previous week. It kept us on track. And it allowed Coye to intervene with valuable insights, both as an outside perspective and as a professor experienced in the area of our work. Our pre-advising meeting served as a great opportunity to recap current pain points, prepare for what we wanted from Coye, and help guide our direction for the next week.

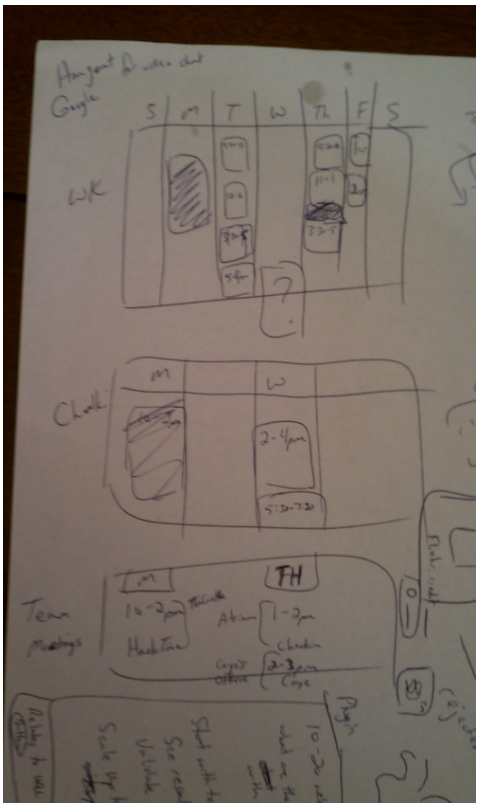


Figure 9. A simple calendar we used to decide when to meet for meetings

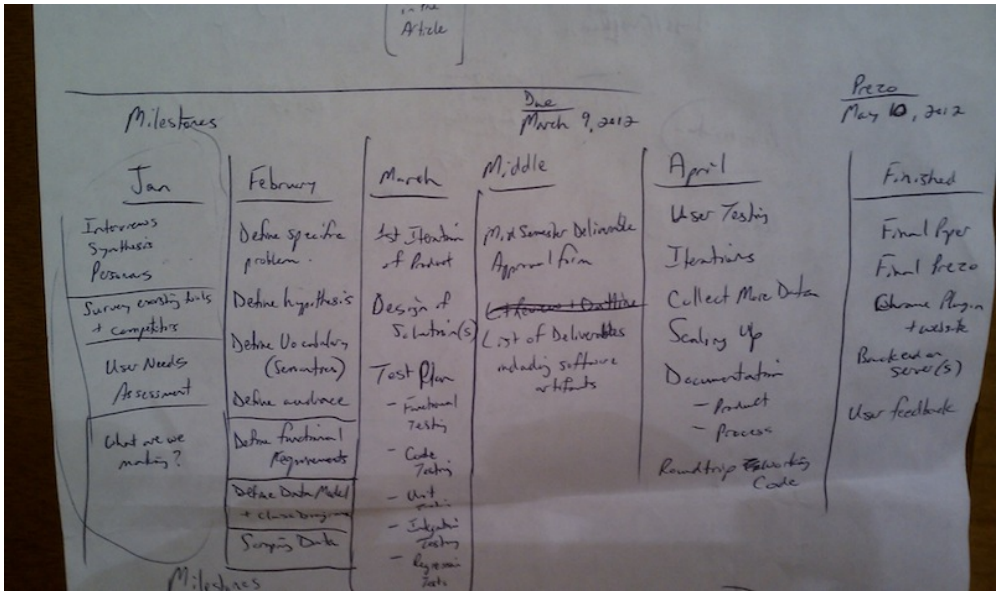


Figure 10. Early on we set out milestones for the project. We met most of our goals.

Collaboration and synergy

Our team divided the work according to our strongest skill sets. Chulki focused on back-end coding and infrastructure. Ram focused on front-end development. Walter focused on interaction design and user testing. At the I School, Morten Hansen taught us about t-shaped people. This is “a mantra that is used at both IDEO and the Stanford d.school to describe people who have both deep skills in one or a few areas and who do their own jobs well and also have the ability to work with others, to share ideas with them, to be be civilized, and contribute to the whole”. [sic]

Collaboration: Morten Hansen's Masterpiece, Mark Bennett, Bob Sutton: Work Matters, 2009-09-14, http://bobsutton.typepad.com/my_weblog/2009/09/collaboration-morten-hansens-masterpiece.html

Our team members are the epitome of t-shaped people. Our group obtained immeasurable synergy by collaborating across skills areas. Frequently the person with the stronger skill in an area served as the leader for the others. And the other team members provided insight and perspective that was invaluable for the leader. Chulki and Ram have overlapping skills sets. Similarly, Ram and Walter have overlapping skill sets. These overlaps offered us the ability to intelligently evaluate and question each other’s design choices.

User Guided Design

Discovery Interviews

During our initial research we sought to validate our initial belief that comments are valuable sources of diverse opinion and new knowledge. To do this we interviewed four people. These interviews were done in an exploratory manner. They began with questions about how the user browses the web. Then they focused on specific aspects of reading articles and comments. The goal was to get an idea of how people approach online discussions. During the interviews with Grant and Jeannette we also had the opportunity to watch them browse articles and comments. We performed a speak-aloud evaluation where they told me what they were doing as they browsed.

Observations from our discovery interviews can be synthesized follows:

People think comments are valuable.
Some people care about the reputation of the commenter.
Some people care about the dialogue surrounding the content.
Some people are looking for an opposing opinion.
People treat comments as different content from the source article.
Since people think comments are valuable, aggregating comments from the entire web (which are relevant to the current source article) is also valuable.

Personas

From our initial interviews we came up with personas. These personas became part of our development process. We weren't designing a solution for just anyone, we were designing for Jane and Juan. If a product feature wasn't for Jane or Juan then it wasn't going to be made. The analogy is that you can make a car for one person because they want a car. And you can make a truck for another person because they want a truck. But if you make an El Camino (a truck car) you won't satisfy either person.

Jane D'oh is an avid reader of online news articles. She browses for articles using the Flipboard application. She has provided Flipboard with a list of publications that she likes. Flipboard then curates a list of articles that they believe she will want to read. When Jane reads an article on Flipboard it is added to their records so that they can offer her more articles like it next time. Jane is interested in knowing how many comments are posted about an article. This gives her an idea of how popular and relevant it is. Occasionally she reads the comments but she's more interested in the number of comments.

Juan Programmer is a tech junkie. He reads about the latest trends and topics in the tech industry. Juan seems to have insight about everything. He habitually reads about product features from as many sources as possible. To do this he searches for a topic using Google and then he reads articles from the top search results. Juan skims comments hoping to find opinions that both validate and contradict his viewpoint.

Prototypes

We began our development process by making prototypes with varying degrees fidelity. For each new feature we developed we repeated the prototyping process.

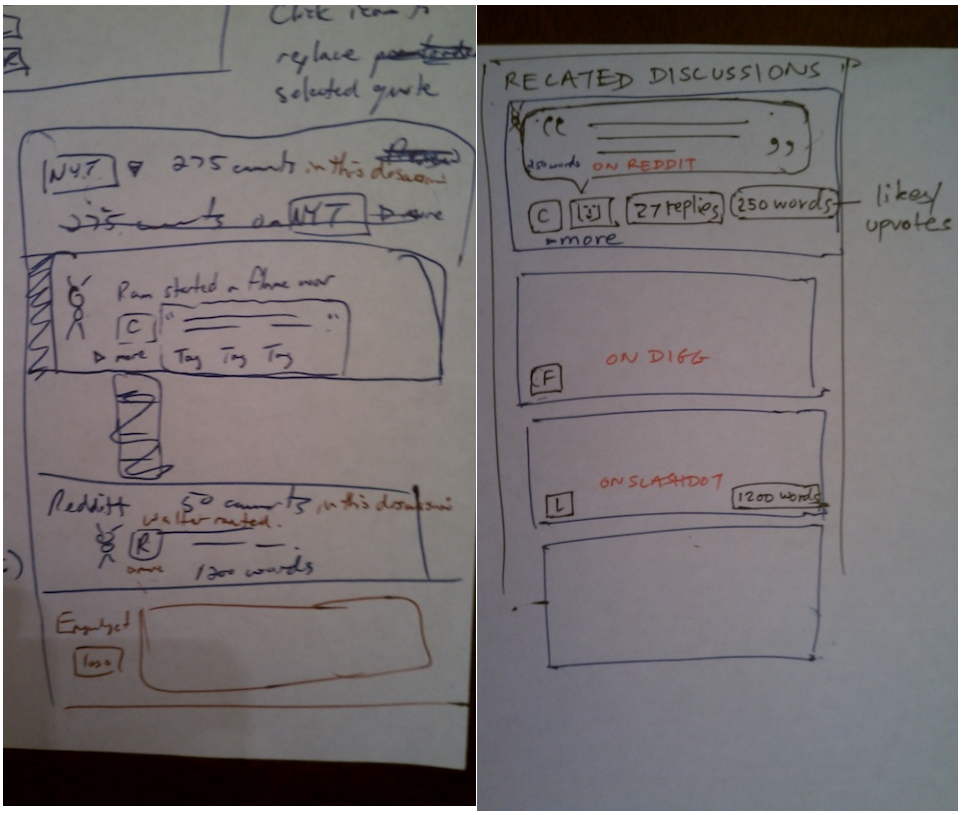


Figure 11. Two of our sketches that lead to higher fidelity prototypes.

We started by drawing sketches on paper based on user feedback and analysis of comment data. The sketches served as a quick and easy way to communicate with the rest of our team. Frequently Walter would draw sketches and show them to Ram and Chulki. We knew we wanted to address information overload and the filter bubble, but initial prototypes helped us decide how to implement a solution. We made the decision to integrate our product into the user's normal reading habits. This aspect became a foundational and critical part of our product. We took every effort we could to provide related articles and comments without asking the user to provide more information.

After the initial drawings we created prototypes with Balsamiq. Balsamiq "Mockups reproduces the experience of sketching interfaces on a whiteboard, but using your computer, so they're easier to share, modify, and get honest feedback on. Wireframes made with mockups look like sketches, so stakeholders won't get distracted by little details, and can focus on what's really important instead." http://www.balsamiq.com/ We were able to export our Balsamiq prototypes to clickable HTML versions by using a script found online. https://gist.github.com/1084424

Our Balsamiq prototypes brought forth new issues. After addressing these issues we wrote a higher fidelity prototype using HTML, CSS and JQuery. These were written as a Google Chrome extension. We knew at this point that we wanted to create a browser extension so we began writing “stubs” that could be connected with data from a server later.

User Testing of Our Browser Extension

We tested our first Chrome extension with five users. We chose interview subjects based on their accessibility and ability to provide insightful feedback regarding our user interface. We wanted to know if the prototype was user friendly, and if the buttons and links made sense. It was a sanity check to see if we were designing appropriate features. We were not.

Sample Feedback

Sample feedback from the interviews is included below. They are not direct quotes. They are notes from the interviewer. Our categorization of the feedback is in parenthesis. Additional notes from these interviews are available in the appendix.

Spell out the filter at the top. **(Usability)**
Scroll bar is invisible to the user. Also might consider putting it on the left side. **(Usability)**
Make comment clickable. Take the user to the target URL (the comment or article). And remove link text. **(Usability)**

Expecting newest comments first **(Consistency and Standards)**
Not sure how comments are sorted initially. **(Consistency and Standards)**
As for discoverability, click comment to read full comment is bad. Better to use ellipsis or “more”. **(Consistency and Standards)**

Can’t vote up/down? **(Interaction with the content)**
Can see comments come from the site with the logo but not sure what article. **(Content choice)**
Comments tab - Thinks of comments as from a user so show thumbnail of user not publication. **(Content choice)**
What’s determining “popular” status? **(Match between system and the real world)**
Pick a word that is more familiar and applicable. **(Match between system and the real world)**

Replies - is it comments that are replies to the article? **(Match between system and the real world)**
Clicking on tabs must open the plugin window. **(Flexibility and ease of use)**

After the initial user testing we rethought the way we presented our navigational features. One glaring issue was that our product had letters representing filters without giving the user enough information to know what the letters stood for. Another issue was with the way we were filtering. We were using words like “depth” to represent the depth of a comment thread. Users we tested didn’t think of depth the same way we did. The filter called “length” had similarly bad reviews. Based off of user testing and further analysis of our data we chose to rethink the filters we were going to include in the product.

We drew more sketches. We made more Balsamiq prototypes. We edited the Chrome extension. And then we tested users again. We interviewed two new users. These notes are also available in the appendix.

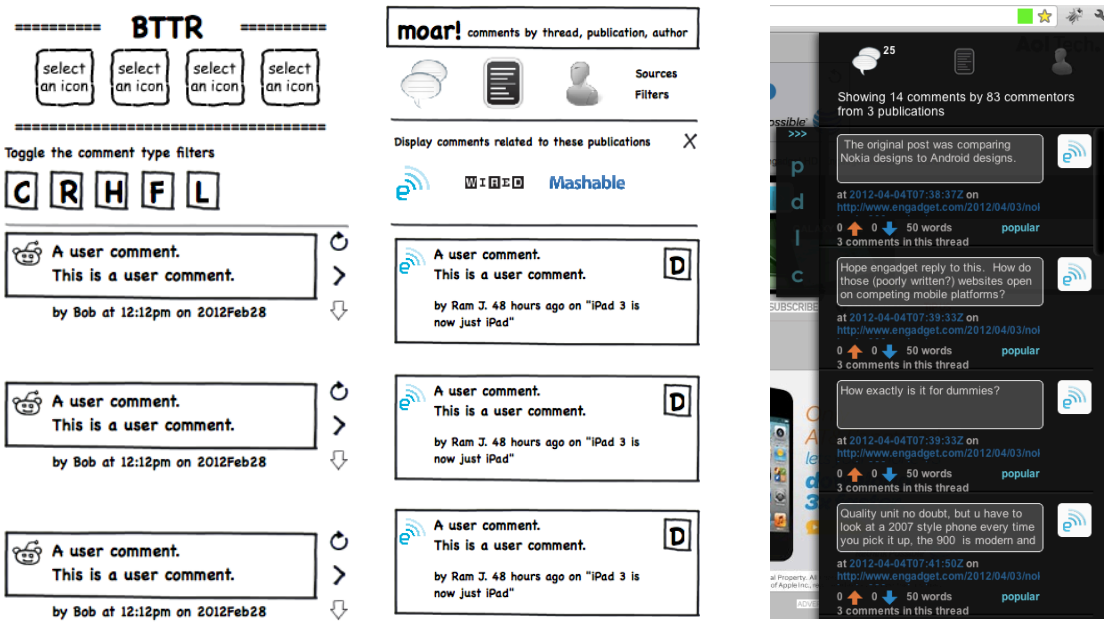


Figure 12. Two of our Balsamiq prototypes and one of our early Chrome Extension iterations.

The second round of user testing validated our changes to the navigation. We switched the filters from the left nav pull tabs to the top nav. We wrote out the names of the filters instead of using the first letter of the word. We used filters that made more sense to the user. And we began using icons for Related Comments and Related Articles in the left nav pull tabs. Users

began to provide positive feedback. They also gave us valuable criticism about how much content we were showing with each resulting comment or article. This lead to a big design feature that we are still using today. This is the link to “view more details” about a comment. Search results have plenty of information but the question became, “How do we show all of it?” With the “view more details” feature we were able to move forward.

We iterated on the filtering mechanisms repeatedly throughout our prototyping and testing. The end result is a product of the information available to us from the websites from where we are pulling content, user feedback, and our own interpretation of what will reveal comments and articles from multiple sources that might offer better discussions about a topic.

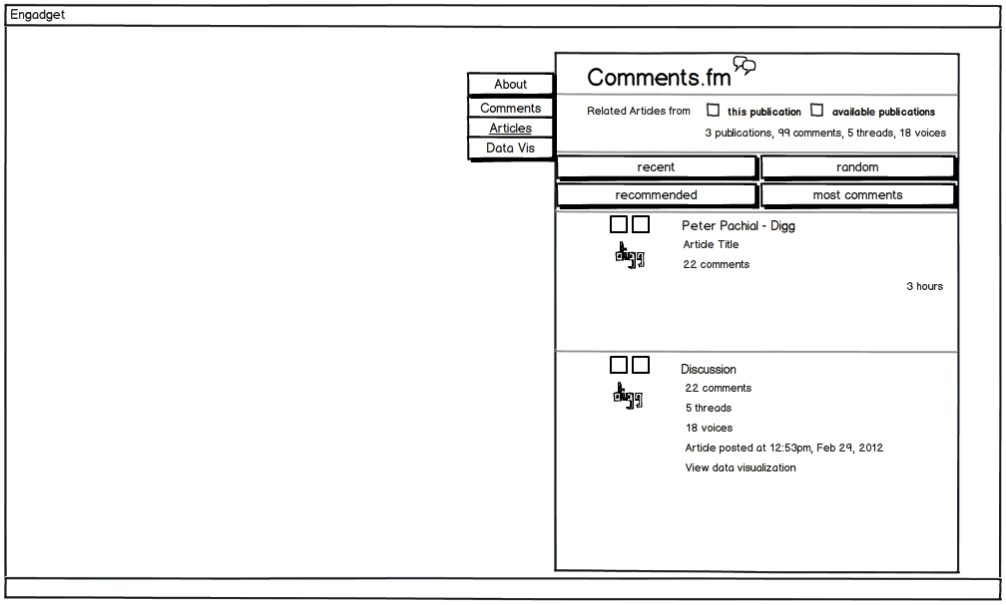


Figure 13. A Balsamiq mockup showing the two panels of information for a comment result.

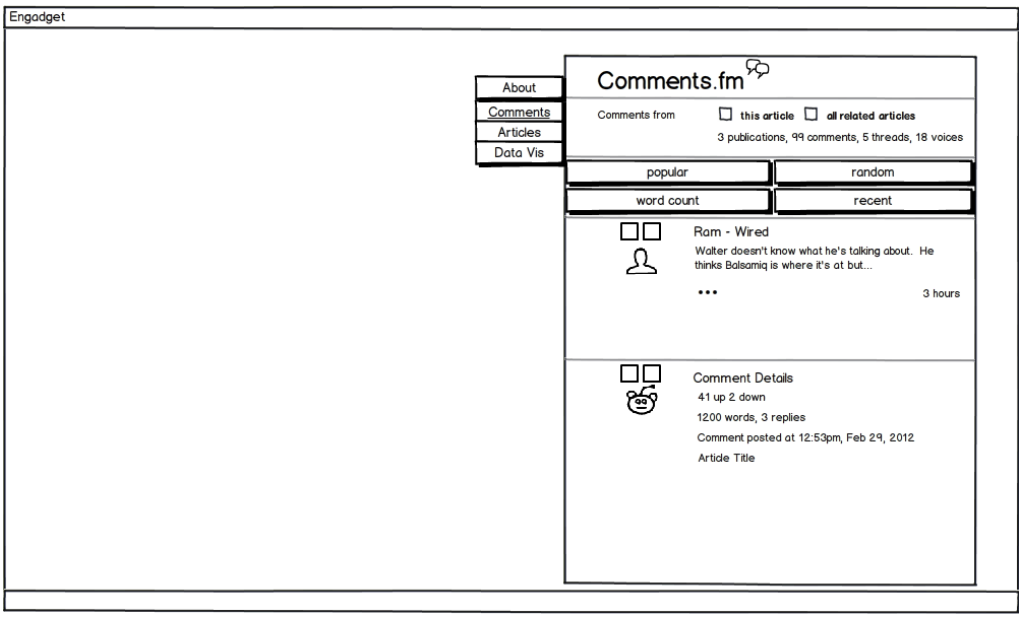


Figure 14. A Balsamiq mockup showing the expand comment feature.

Backend: Data

Scraping

Crawlers

To collect articles and comments on the Web we built crawlers. They scrape web pages, extracting articles and comments from them. We surveyed the current available solutions and found that they were not suitable for our project.

For instance, general-purpose crawlers, which aim to collect web pages, cannot extract objects within pages. In our case the objects were articles and comments. They usually provide ways to post-process contents but we found that they were not flexible enough for our purposes.

In other case, structure-detecting crawlers are designed to detect objects from crawled pages but they are still in the research or development phase. We found that they are not ready for reliable use.

Therefore we built three types of crawlers.

RSS crawlers: Most websites provides RSS feeds which contain various information of recent articles, such as title, published timestamp, etc. Therefore we built an RSS crawler which fetches and parses RSS feeds of articles.

However, most feeds contain only recent articles, not past article information. In addition, some feeds include only the summary of articles, not the full text of them. In such cases, finding similar articles may not work as well as when the full text is available. Therefore, it is required to parse HTML web pages.

Site-specific crawlers: A site-specific crawler processes common HTML structure on specific web site. For example, MashableCrawler parses and extracts articles and comments on web pages of Mashable.

One advantages of implementing site-specific crawler is that it allows to collect site-specific data, such as up-down votes, user profile etc.

There are several drawbacks, too. First, a crawler has to be implemented for each site. Second, even a web site may have different HTML structures. For example, we found that Wired has a slightly different HTML structure in their main website, Cloudline and Webmonkey. In addition, Webmonkey even has a different domain name, although articles are served in Wired's RSS feed. Third, if a website changes its HTML structure, crawler will be broken. In such cases, RSS crawlers may be used for collecting articles information until site-specific crawler is updated.

Disqus crawler: Disqus is currently one of the most popular commenting platform on the web. As of 2011, Disqus had nearly 500 million unique visitors every month, and 75% of websites who use a third party commenting system used Disqus.

Source: The Numbers of Disqus, danielha, Disqus Product Blog, 2011-04-05, http://blog.disqus.com/post/5192492910/the-numbers-of-disqus

And two of our first target websites were using Disqus. To collect comments on more websites, we implemented Disqus crawler, which uses Disqus API to collect comments on articles.

As a result, currently we support three websites in the following ways:

- Mashable
    - Get new articles from the RSS feed
    - Visit the article page
    - Get comments on the article using a site-specific crawler
- Engadget / Wired
    - Get articles from RSS feed
    - Visit the article page and get Disqus information of the article
    - Get comments by using the Disqus crawler to get the information

Crawling strategies

We use two strategies to achieve real-time service: periodic and on-demand crawling.

First, ***periodic crawling new articles*** is to check RSS feeds or article list pages in a certain time interval. By fetching articles and comments periodically, regardless of user's requests, we can prepare data to serve in advance.

Second, ***periodic crawling outdated articles*** is to re-crawl articles and their comments when it has been certain amount of time past since they were crawled.

Third, ***on-demand crawling*** is to fetch articles and comments on them when a user request information of them. It allows the server to collect and process requested articles right after the user's requests.

In the next step, we plan to improve crawling strategies using adaptive crawling. For instance, the frequency of publishing new articles for each site can be estimated from past trends. In addition, if an article is requested a lot in given times or comments are made on the article often, we need to update articles and comments more often than other articles.

Storing

NoSQL

We use MongoDB http://www.mongodb.org/ to store collected articles and comments. MongoDB is a document-oriented NoSQL database system. Since metadata of articles and comments are different between websites, relational database systems which require a pre-defined schema is an appropriate solution. For instance, we have ten fields for Disqus-specific data in the comment model. Document-oriented storage like MongoDB does not require pre-defined schema and each document may have a different schema, which is required in our case. If we were to use a relational database management system (RDBMS) we would need to alter table schemas whenever new sites have custom metadata. This is an undesirable situation.

Indexer

We use Apache Solr (http://lucene.apache.org/solr/) to index collected data. Once a crawler finds new content, articles or comments, that content is passed to the Solr server.

Stored data to date

As of May 3, 2012 we have stored the following data, listed by website.

Website	Articles	Comments
Engadget	730	203387
Mashable	345	3873
Wired	193	5151
Total	1268	212411

Analyzing

Graphs

To find patterns from data, we created various graphs using R. For instance, the following graph represents the number of comments on the fourth most commented article. It has 12554 accumulated comments over time.

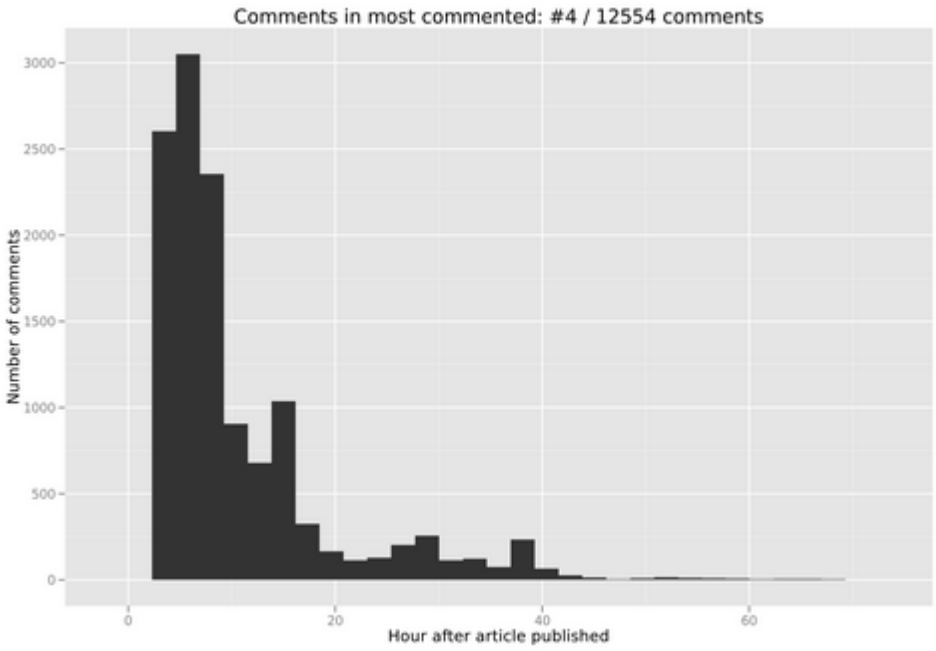


Figure 15. Graph of comments produced in R.

The following boxplot graph shows the distribution of the number of comments on each article by source.

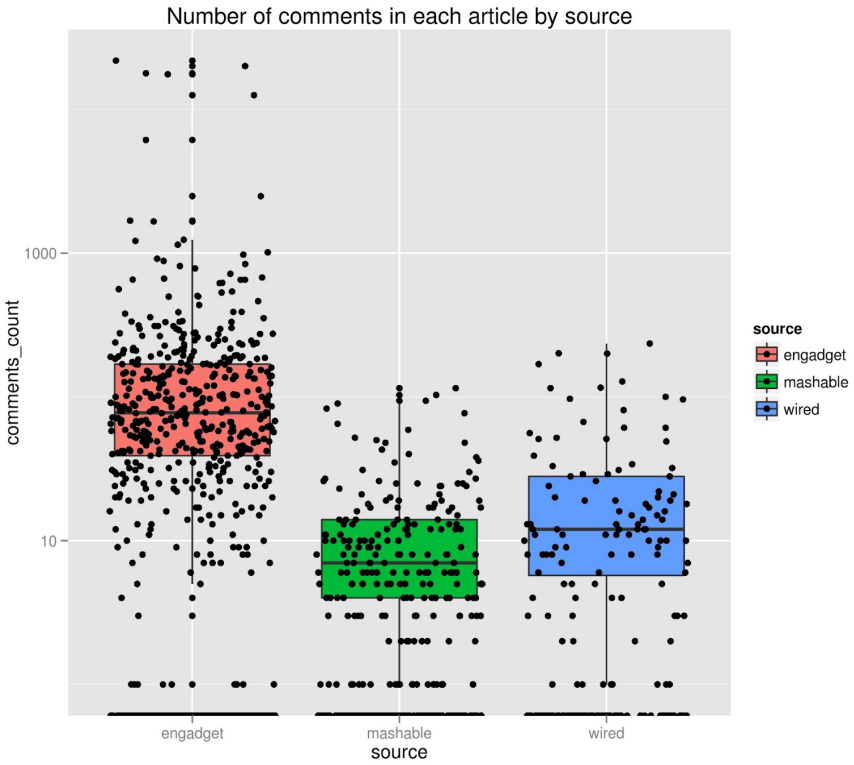


Figure 16. Graph of comments by source produced in R.

Indexer

Solr provides several useful out-of-the-box features, including searching by queries or finding similar documents to a given document. For instance, Solr provides stemming words, removing stopwords and mapping characters.

After testing we found that we needed to change some configuration. For instance, we found that "Wired" is not treated properly because it is considered as a past tense of "wire." Therefore we added "Wired" to a list of word protected from stemming.

Serving

API server

We created an API server built on Django. It provides three categories of APIs. To support our various needs each API call accepts parameters, such as keyword, article url, number of result,

or offset.

- Dataset
  - /stats
- Articles
  - /articles/recent
  - /articles/search
  - /articles/lookup
  - /articles/[id]/comments
  - /articles/[id]/related_articles
- Comments
  - /comments/search
  - /comments/random
  - /comments/on_related_articles
  - /comments/[id]/subcomments

JSON response

Currently the API server provides JSON(JavaScript Object Notation) responses. JSON is a lightweight and flexible format. Most API providers provide it. The following is an example of a response to /comments/random.


```
{
  "comments": [
    {
      "body": "Silly Verizon....\n\nTricks are for kids!",
      "wordcount": 6,
      "site": "engadget",
      "down": 0,
      "article": {
        "author": "Brad Molen",
        "url": "http://www.engadget.com/2012/04/11/verizon-upgrade-fees/",
        "posted_at": "2012-04-11T15:32:00Z",
        "title": "Verizon pushes its upgrade fee to $30 on April 22nd",
        "site": "engadget",
        "comments_count": 541,
        "id": "4f85a536d82d805a55001b07"
      },
      "subcomments_count": 0,
      "author_avatar_url": "http://disqus.com/api/users/avatars/svargas05.jpg",
      "id": "4f85c15ed82d805a55003c14",
      "author": "svargas05",
      "url": "http://www.engadget.com/2012/04/11/verizon-upgrade-fees/#comment-
494484603",
      "posted_at": "2012-04-11T17:12:15Z",
      "up": 3,
      "highlighted": false,
```

```
        "depth": 0,
        "article_url": "http://www.engadget.com/2012/04/11/verizon-upgrade-fees/",
        "author_url": ""
      }
    ]
}
```


Challenges

Blocking request: Since fetching and processing data on the fly takes time, we use Celery(http://celeryproject.org/), an asynchronous task queue system. By using it clients can receive an immediate notification that the requested data is being crawled, instead of being blocked until the fetching task is done.

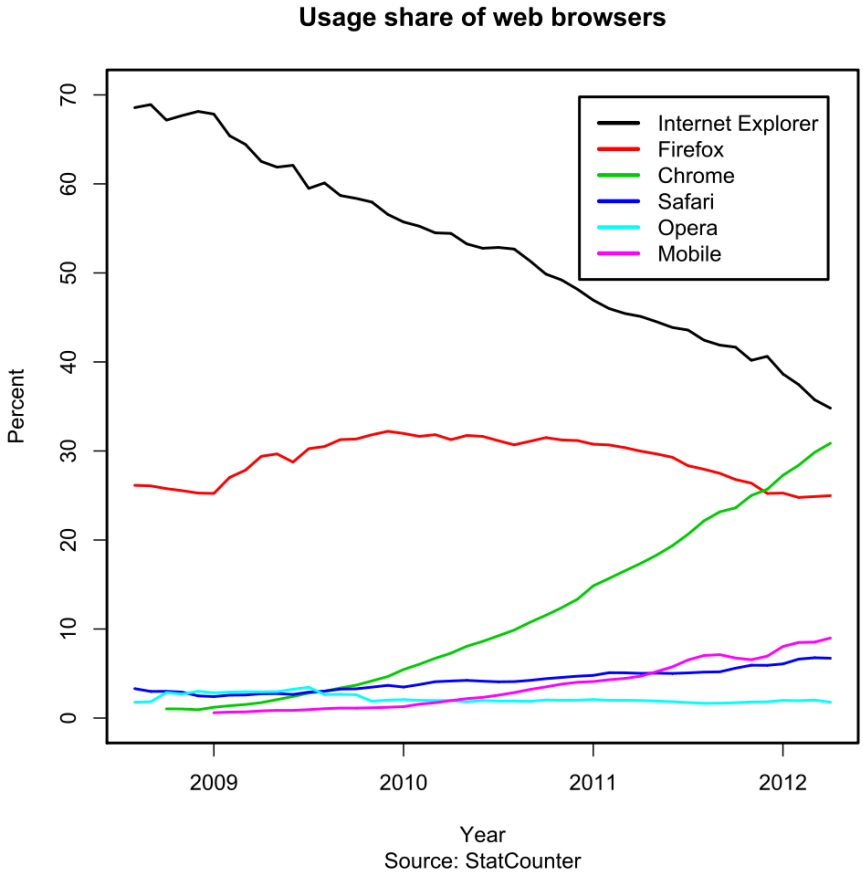
Chrome Extension

Justification

The Chrome extension is the user facing component of our product. Our solution works within the context of a user's browser activity. The obvious choice for implementing such a solution is with browser plugins or extensions. Of all the popular browsers in the market, Chrome was the clear winner for this project for the following reasons.

Popularity

Chrome is the second most popular browser in the world and growing at a remarkable rate. Internet Explorer (IE) - including all versions - is the most popular browser. However version 15 of Chrome had a 24.55% of the worldwide market edging out IE8's 22.16%



Source: Chrome 15 Beats Internet Explorer 8 as World's Most Popular Browser, Peter Pachal, Mashable, 2011-12-15, http://mashable.com/2011/12/15/chrome-leapfrogs-ie8/

Target market

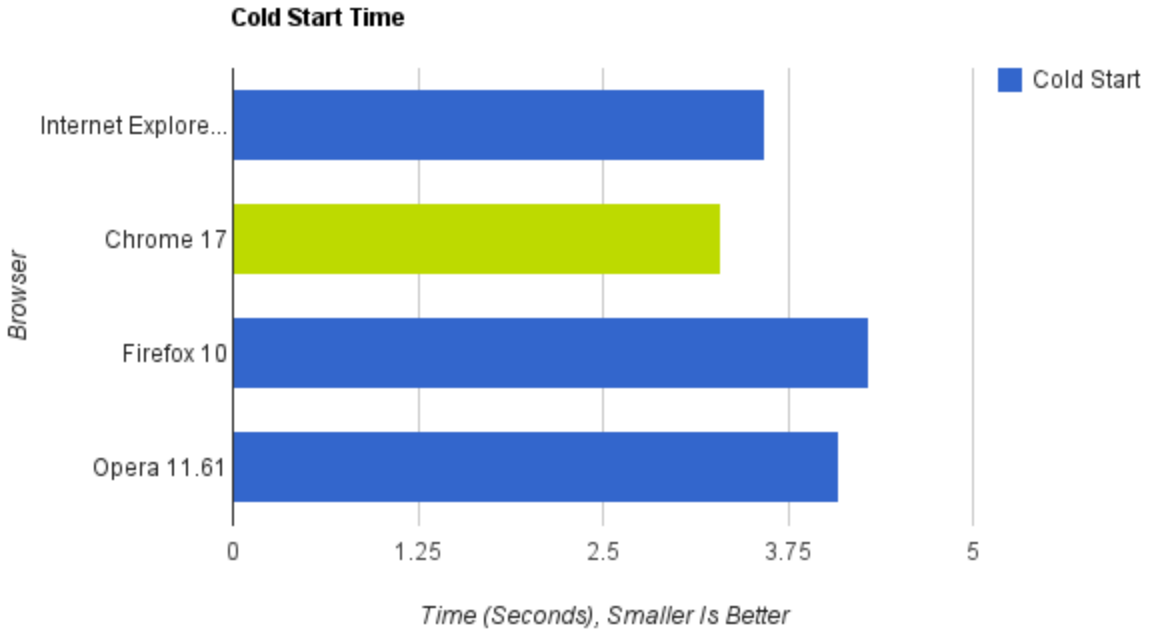
Chrome is the top browser of choice for our target users. Among Mashable readers it's by far the most popular browser at 27.9%. Firefox is second at 22.4% and IE is in third place with 19.4%. (See figure) These numbers also include mobile browsers. Engadget and Wired readers also largely prefer Chrome.

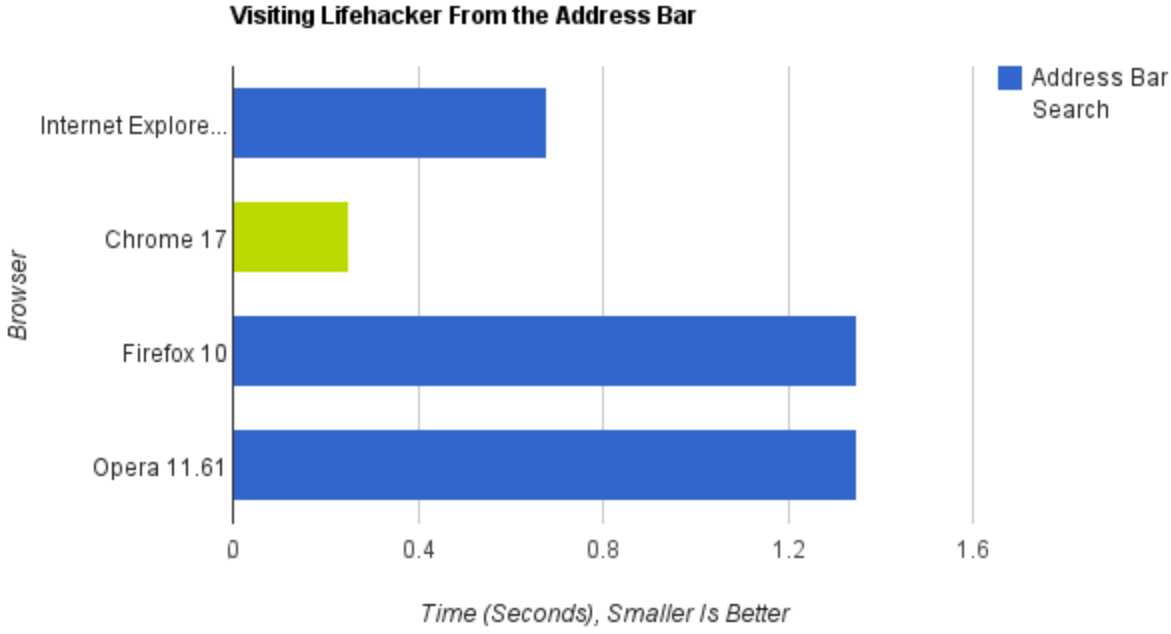
Development

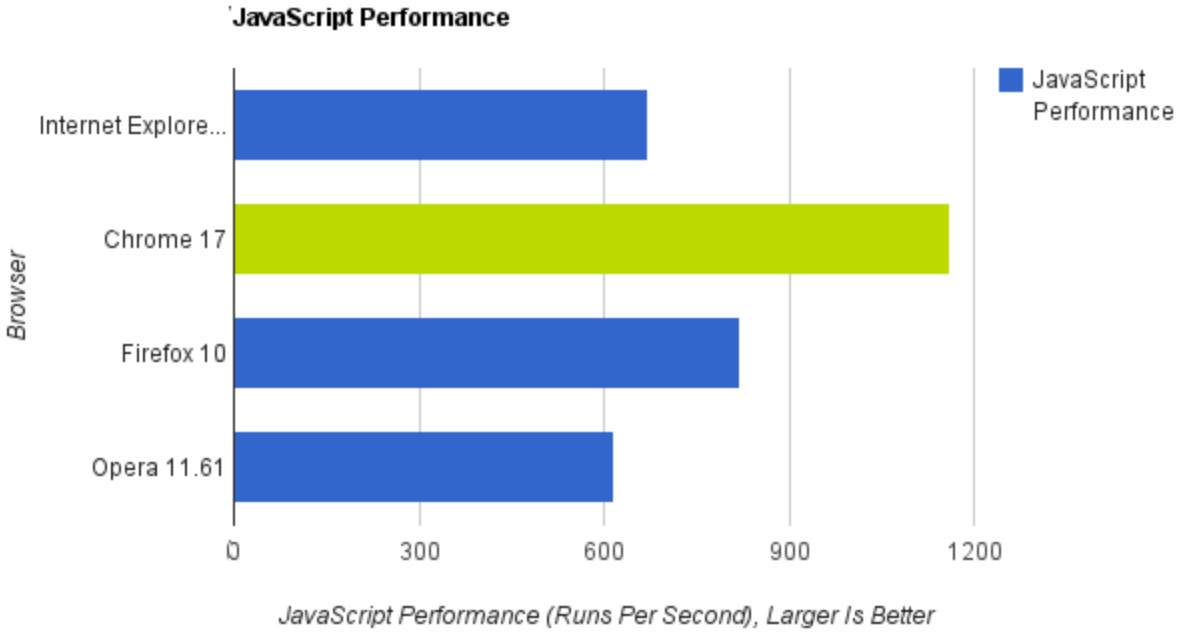
Chrome extensions are written entirely with web development languages such as Javascript, HTML and CSS and are easy to build, maintain and deploy. Use of pure web technologies means that extension features can be easily extended to a website in the future.

Performance

Chrome has consistently been a high performance browser. Its rendering engine (webkit) and Javascript engine (V8) are winners of several benchmark tests. Performance was a key factor for choosing Chrome as the browser platform for our product. Our product works by injecting and rendering a fairly cpu and memory intensive DOM into one or more tabs open in a user's browser. Although Chrome has recently been lagging in memory efficiency compared with Firefox, it starts up fast, has a fast url load speed, and a high performance Javascript engine, making it an ideal choice for our product.

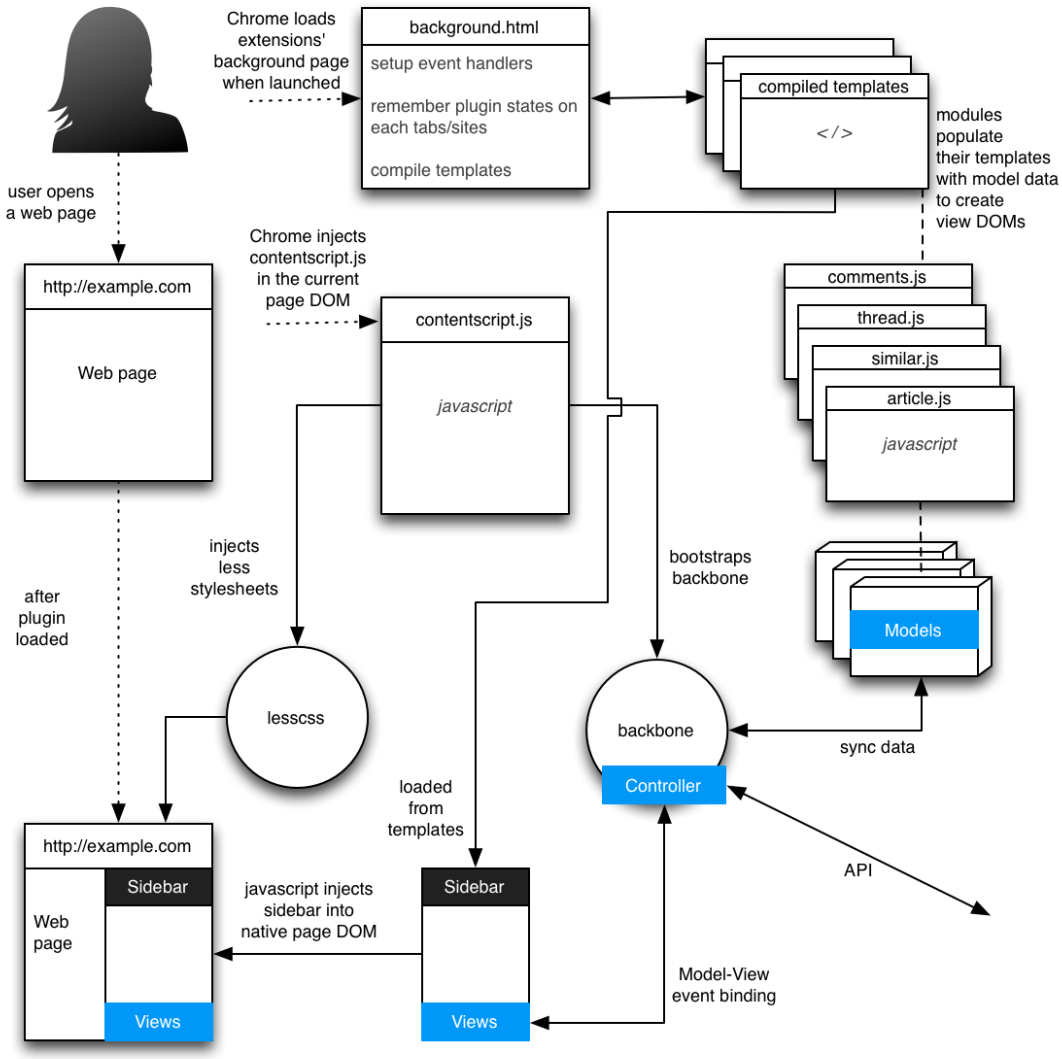

Cold Start Time
Internet Explore...
Chrome 17
Firefox 10
Opera 11.61
Browser
0
1.25
2.5
3.75
5
Time (Seconds), Smaller Is Better
Cold Start


Visiting Lifehacker From the Address Bar
Internet Explore...
Chrome 17
Firefox 10
Opera 11.61
Browser
0
0.4
0.8
1.2
1.6
Time (Seconds), Smaller Is Better
Address Bar Search



Source: Browser Speed Tests: Chrome 17, Firefox 10, IE9, Opera 11.61, Whitson Gordon, LifeHacker, 2012-02-15, http://www.lifehacker.com.au/2012/02/browser-speed-tests-chrome-17-firefox-10-internet-explorer-9-opera-11-61/

Architecture



Challenges

Code Organization

Javascript has no recommended or well known code organization principles. The language does not explicitly favor use of classes and modules for organizing code. The Chrome extension

has about 800 lines of javascript code. We use closures and a memoizer for separating different sections of the extension into modules.

The memoizer creates a namespace for all the modules also prevents access to the modules outside the namespace, effectively creating a local scope for the modules.

```
// a memoized module loader/cacher based on http://weblog.bocoup.com/organizing-your-backbone-js-application-with-modules
var bttr = {
    api: 'http://grepvine.info/api/v1/',
    filter_map: {
        threads: { popular: 'popular', random: 'random', wordcount: 'wordcount', replies: 'active'},
        articles: { commented: 'comments', random: 'random', relevant: 'relevancy', recent: 'recent'}
    },
    filter_stat_map: {
        threads: { 'popular': 'up', 'wordcount': 'wordcount', 'replies': 'subcomments_count'},
        articles: { 'commented': 'comments_count', 'relevant': 'score', 'recent': 'posted_at' }
    },
    module: function() {
        var modules = {};
        return function(name) {
            if(modules[name]) {
                return modules[name];
            }
            return modules[name] = {Views: {}};
        };
    }() // http://benalman.com/news/2010/11/immediately-invoked-function-expression/
};
```

XHR restrictions

The Grepvine Chrome extension works by injecting its DOM into the the current web page open in the browser tab. There is no redirecting or mangling of the URL visited by the user. This makes every request made to a server from the extension appear to be originating from the host domain. For instance, if a user visits a page on mashable where Grepvine shows up, all requests made by Grepvine to the server appear to be originating from mashable. This creates problems for accessing local static resources within the extension, since all relative URL's point to resources on the mashable domain. This prevents us from linking to static resources such as images from CSS files or even inline HTML in templates. All local resource links must be resolved through the Chrome extension api.

CSS isolation

This problem is closely related to XHR restrictions. Following is an excerpt of a bug from our issue tracker.

> Context:
> Chrome extension contentscript.js runs like a regular script inside a web page.
> We are using this to modify the DOM of the page to paint our sidebar.
>
> Problem:
> Since our sidebar is not isolated from the host page, the parent styles can affect our sidebar.
> This is bad because the sidebar may look slightly different on each page (wired vs engadget for instance).
>
> Bad/failed solutions:
> Showing the sidebar inside an iframe would isolate its css perfectly
> 1. If iframe is created by inserting DOM without a src = url:
> Scripts fail because of XHR restrictions (for instance less.js running in the iframe cannot access the local styles.less inside our plugin since chrome only allows XHR to http:// and our local extension files are on chrome-extension:// (similar to file://). We could host the styles and libs on the server, but then the user will fetch them from the server on each page load which sucks.
> 2. If iframe is is inserted with src = url:
> contentscript.js cannot even access the DOM or even the iframe window because of chrome restrictions.
> http://code.google.com/p/chromium/issues/detail?id=20773
> http://code.google.com/p/chromium/issues/detail?id=70866
>
> tldr:
> Isolating extension style from web pages is only possible with iframes (afaik) and iframes do not work well in chrome's sandboxed environment for extensions.
>
> Current approach:
> Deal with styling quirks on a per site basis or as weird behavior is noticed.

Product Design

Overview and Features

Our final product is a culmination of a semester's worth of prototypes, user testing, analysis and iterations. Our goal was to address the problem of information overload and the associated search bubble. Existing solutions offer content from multiple sources delimited by past user behavior. We created a product that pulls information from multiple sources and uses crowdsourcing features more commonly associated with news aggregation websites. We index our content regularly. We apply TF/IDF, stopwords, mapping characters, and stemming to our indexed body of data.

Our product identifies articles related to the article a user is browsing. The user is provided with the most relevant comments and articles from our algorithm. We took every effort to be sure the user doesn't have to leave the article they are on to see this related information. This means that a user will use our product to find related articles and comments from the same they are already reading an article. This is an intentional design choice.

Branding

Figure. The header logo within our extension, the related comments tab icon, the about us tab icon, and the icon for the Chrome Web Store..

The roots of the name Grepvine derive from an association with a cluster of grapes that are connected by a vine. An article and comments forms a cluster. The content of the article and comments that a user has browsed to provides a vine, or common facet, that connects a user to related clusters. Our logo displays a node and edge pattern with a high resemblance to that a cluster of grapes.

Navigation

Our product is a Google Chrome extension. Features include an icon in the location bar that allows a user to toggle the extension on and off for the URL address they are currently on.

When the extension is enabled and the user browses to an article on a supported website they will see left navigational pull tabs for our product. The pull tabs include the following icons: an arrow, a comment bubble, a page of text, a chart, and a small piece of the Grepvine logo. A user may click on any of these icons to view the extension. A user may click on the arrow to hide the extension. The comment bubble icon also displays the number of related comments that are available in our result set. The page of text icon also displays the number of related articles from our result set.

The related comments page shows a list of comments, the avatar of the user who wrote the comment, their username, the source of the article, the comment, and various other information about the comment. We currently display a subset of 25 related comments.

The related articles page shows a list of articles, the thumbnail of the article's source, the author of the article, the article title, and when it was posted. We currently display a subset of 25 related articles.

Filtering and Sorting

On the comments page a user may choose to include comment from all sources included in our corpus of data, or they may choose to include only comments from the current article they are reading. This feature is intended to give the user access to our sorting features on the current article's comments when they may not be available on the original site.

Sorting features vary slightly on the comments page from those available on the articles page. On the comments page we include the ability to sort by facets identified by the names "popular", "wordcount", "replies", and "random". "Popular" refers to the number of upvotes a comment has received at the original source of the comment. "Wordcount" refers to the number of words in the comment. We have chosen to display comments in descending

order to capture the idea that longer posts are indicative of a user having something of substance to say. “Replies” refers to the number of a replies a comment has received. This is common on websites that allow threaded discussions. “Random” is a random related comment from the available corpus. The purpose of including a random option is to increase the likelihood of reading a comment that outside of a user’s filtering bubble, while also keeping in line with being related to the article they are reading.

On the articles page we include the ability to sort by facets identified by the names “relevant”, “commented”, “recent”, and “random”. The default is “relevant” articles. These are articles that are deemed to be most relevant by our indexing algorithm. “Commented” refers to articles with the most comments, displayed in descending order. “Recent” displays the most recently posted related articles first. “Random” refers to a random list of related articles from our relevant result set. This is available as a way to disrupt the filtering bubble associated with curated content results.

Visualization

Statistics

We have included brief statistical information about our relevant result set. This includes the last time an article has been indexed. It also includes the total number of related comments and related articles in the result set.

Dynamic Visualization

A dynamic visualization is available for each article. The visualization allows a user to see the rate at which comments are placed within the first four hours after an article is posted. It also displays the depth of the comment in the comment thread, and the comment itself. The visualization is available on our web page, not as part of our browser extension.

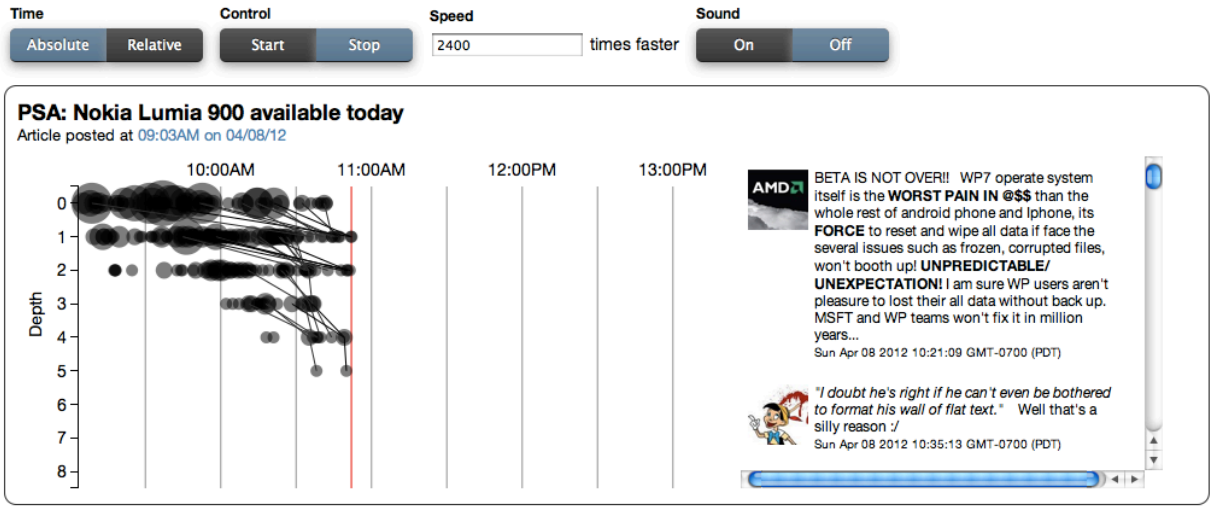


Figure. Our visualization of the comments for an article over the first four hours after it was posted.

Goals and Findings

We set out to improve upon the status quo related to reading online content. We saw problems associated with information overload and a search bubble. Users ask Google for search results related to the topic they are interested in. These results are frequently skewed by the user's prior browsing behavior. Users also seek curation services, like Flipboard, to help give them content that they might find interesting. This content is skewed by the publishers best interests.

We have developed a solution that we believe addresses these problems. Our Google Chrome extension integrates seamlessly into user's existing online news reading behavior. Content is processing for relevance without the user having to do anything, and the results are amazing. We have helped users avoid the search bubble. We have also opened up a new discussion about how contents may be treated as a new corpus of information. All of this while reducing information overload with relevant related content.

Future

Product

Our product has been written with clean, readable, abstracted code. It is flexible enough to handle additional sources of content from online publications using comment platform APIs or RSS feeds. Any publication that can be scraped is also a candidate for our corpus. One aspect that we intend to improve upon is our indexing algorithm. The goal is to provide even more relevant related content.

A feature that we may decide to build involves Facebook integration. If we integrate Facebook Connect into our product users could see comments on related articles that are posted by people they know.

Legal

We have made this product for a school project. If the project were to continue we would consider getting official legal advice. As we understand it we are legally allowed to scrape the content we are using. We are presenting comments that are already in the public domain, and there is no copyright protection for these comments. We will need to review further the sites we have included to know for sure if we have violated any terms of service agreements. Upon request we will cease and desist.

Business Aspects

We see a business use case for Grepvine. If a publisher has many web properties they may want to refer users to content from one property to another. Our product could assist the publisher with this value proposition. More specifically, we could build into our product the ability to track click behavior and offer a publisher insight into which articles their users are reading.

We learned in our Managing for Information Intensive Companies course about the difficulties large organizations have with innovating. Our product is something that may not fit into the normal business strategy of a large media organization. We have the benefit of not having to

report to a manager or think about a bottom line while we develop. Furthermore, we are not tied to a specific client. We are free to innovate as we please. A large media organization may see us as an acquisition target to acquire our technology.

Appendix

Terms Glossary

Google Chrome Extension Installation

Grepvine Product Walkthrough

Grepvine Color Palette

Survey of discussion systems on websites

Server Diagrams

Browser Extension Diagrams

Discovery Interviews

Findings from user testing grouped by interview

Findings from user testing grouped by topic

Terms Glossary

Key Terms

Articles
: Articles are written by authors. They are posted to publications.

Comments
: Comments are posted by a commenter. They are posted to an article.

Publications
: Publications are websites. They include Engadget, Mashable and Wired.

People
: Comments are posted by comment authors (aka commenters)
: Articles are written by authors

Filters

Comments

- Popular: Comments with the most upvotes [ordered by most upvotes, descending]
- Replies: Related comments with the most replies [ordered by most active, descending]
- Wordcount: Related comments sorted by the number of words in the comment [ordered by most words, descending]
- Random: Random list of related comments from our indexing engine

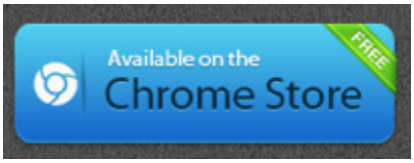
Related Articles

- Recent: Listing of all articles related to the article [sorted by most recent articles first, descending]
- Comments: Articles with the most comments [sorted by most comments, descending]
- Relevant: Articles the indexing engine recommends
- Random: Random list of articles related to the article

Google Chrome Extension Installation / Uninstallation

Instructions to install the extension

1. Go to http://Grepvine.info and click on the download button, or
2. Download and install the extension from the Chrome Web Store
    a. Available in the Chrome Web Store https://chrome.google.com/webstore/category/home
    b. Search for Grepvine
    c. Click on the button to install the extension



Instructions to uninstall the extension

1. Click the wrench icon  on the browser toolbar.
2. Click **Tools**.
3. Select **Extensions**.
4. Click **Uninstall** for the extension you'd like to completely remove.

Or you can also temporarily turn off an extension by disabling it on the Extensions page.

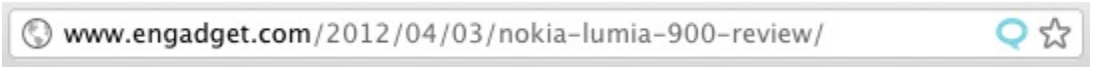
Grepvine Product Walkthrough

1. Browse to a Grepvine indexed article
2. Interact with Grepvine pull tabs
3. Interact with the location bar icon
4. View related comments
5. View related articles

1. Browse to a Grepvine indexed article

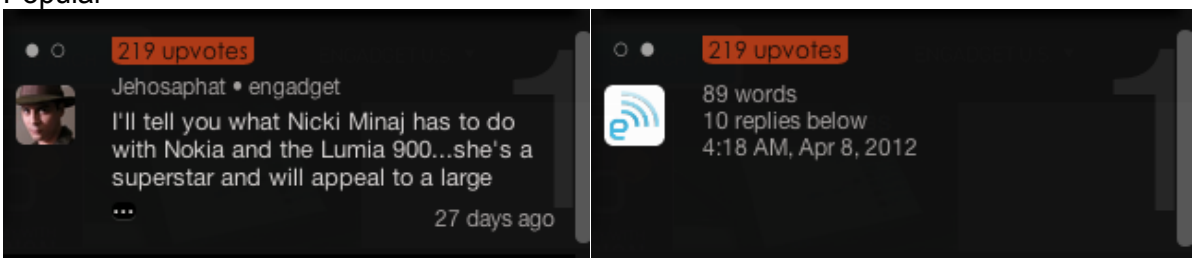
2. Interact with Grepvine pull tabs

3. Interact with the location bar icon

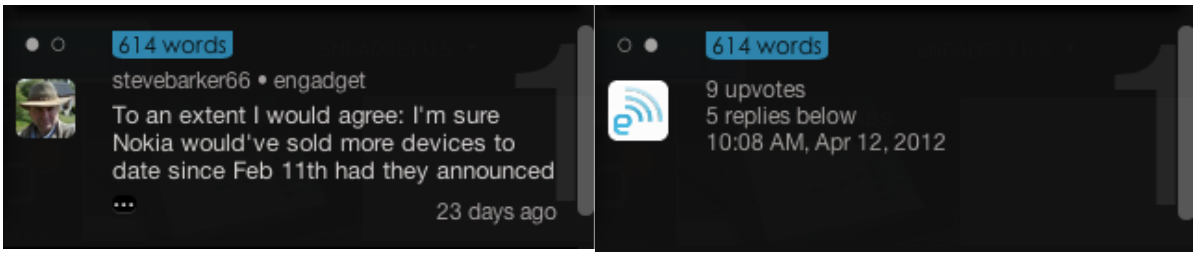


4. View related comments

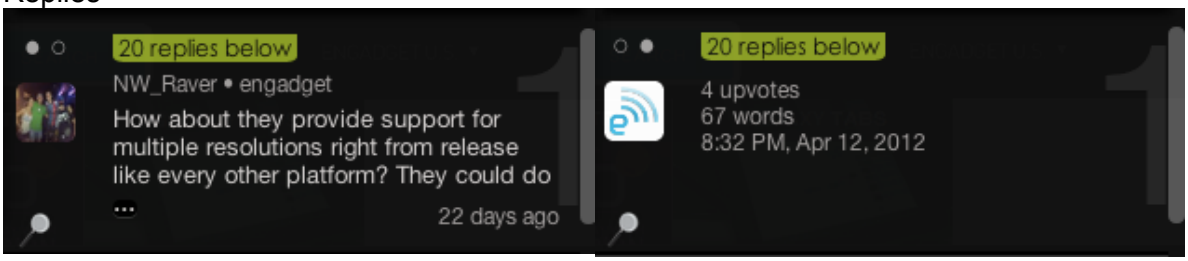
Popular



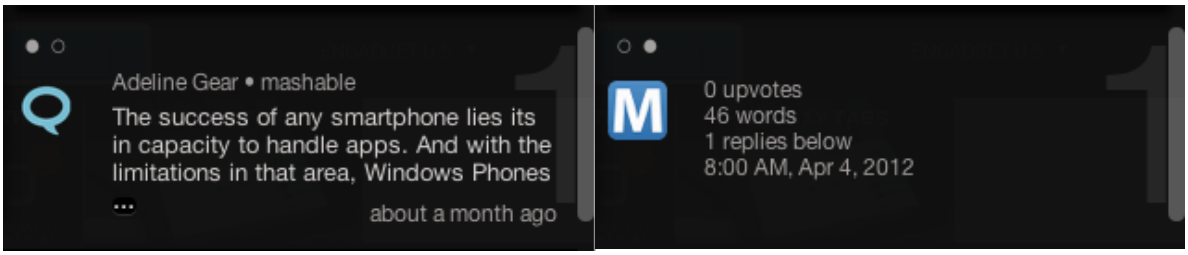
Wordcount



Replies

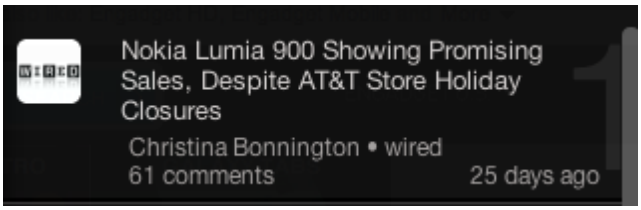


Random

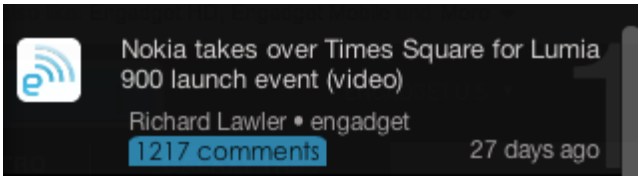


5. View related articles

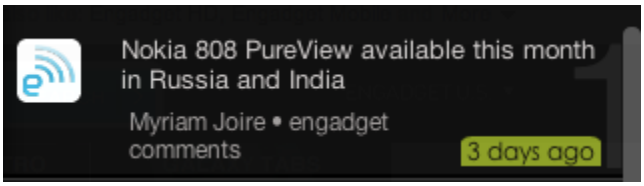
Relevant



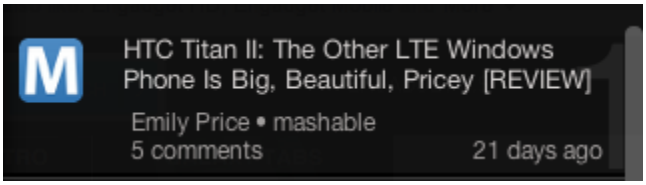
Commented



Recent



Random



Grepvine Color Palette

Logo

Background:	#000000
Text:	#43C6DB
Graphic:	#43C6DB

Comment sorting icons

Popular:	#BB2904
Wordcount:	#0484B0
Replies:	#87A812
Random:	#922A84

Article sorting icons

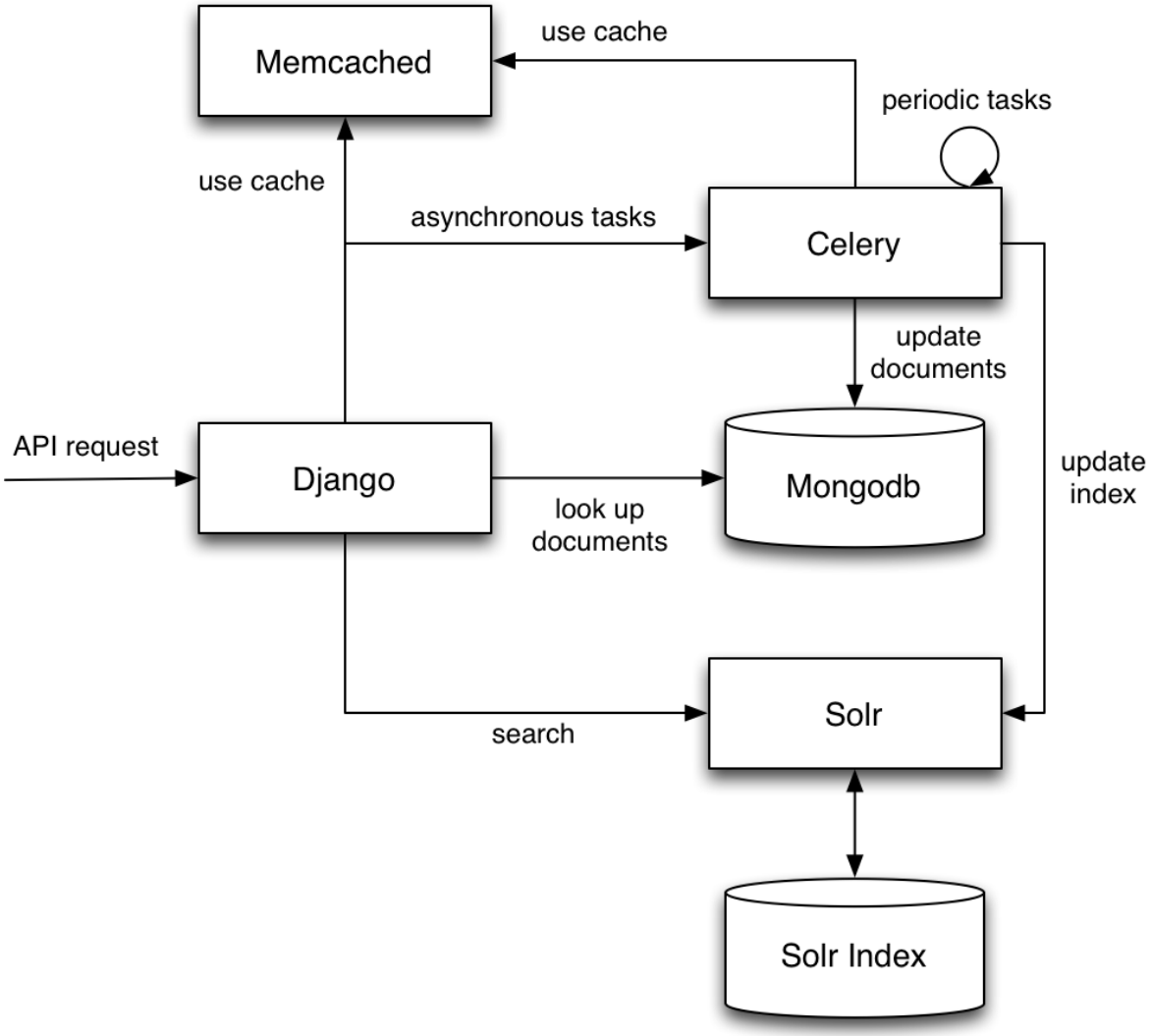
Relevant:	#BB2904
Commented:	#0484B0
Recent:	#87A812
Random:	#922A84

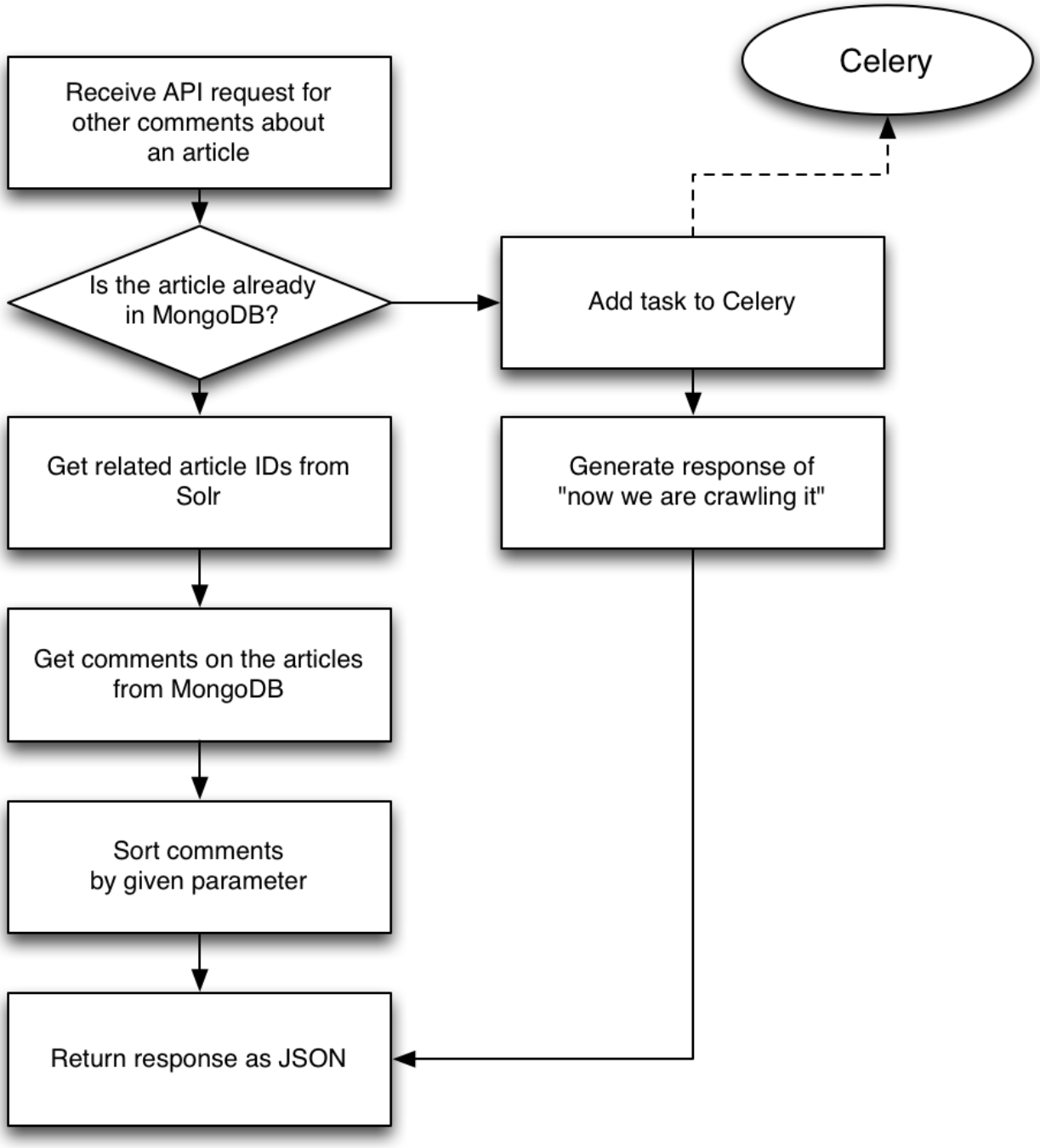
Survey of Discussion Systems on Websites

Website	System	Feedback	Thread	Commenter Info	Auth	Sort by	Note	Alexa Rank
Mashable								
TechCrunch								
Engadget								
Wired								
Ars Technica								

Website	URL	System	Feedback	Thread	Commenter information	Auth	Sort by	Note	Alexa Rank
Mashable	http://mashable.com/	Own system	Thumb up	?		Facebook, Twitter	(only oldest first)		201
TechCrunch	http://techcrunch.com/	Facebook comments	Like	two level		Facebook, Yahoo, AOL, Hotmail	Social Ranking, Chronological, Reverse Chronological		300
Engadet	http://www.engadget.com/	DISQUS	(same)	(same)		(same)			364
Gizmodo	http://gizmodo.com/	Gawker						promoted by ??	628
Wired	http://www.wired.com/	DISQUS	(same)	(same)		(same)	Popular, Best rating, Newest, Oldest		631
lifehacker	http://lifehacker.com/	Gawker	(none as anonymous)	?		?	(only newest first)	can add image url or youtube link	671
ars technica	http://arstechnica.(	Own system	(none as anonymous)	?		Own account	(only oldest first)		1764
GigaOM	http://gigaom.com/	Own system	(none as anonymous)	three level		Anonymous, LinkedIn, Facebook, Twitter, Wordpress.com, Own account	(only oldest first)		2158
ReadWriteWeb	http://www.readwriteweb.	DISQUS	Like	more than four		DISQUS, Google, Twitter, Facebook, Yahoo, OpenID	Popular, Best rating, Newest, Oldest	It also displays recent tweets containing the link of the article in "Reactions" section.	2244
TUAW	http://www.tuaw.com/	AOL???	Up/Down	two level?		AOL, Facebook, Google, Yahoo, Twitter	Newest, Oldest		5502
Techmeme	http://techmeme.com/								7163
Digg	http://digg.com/								189
Reddit	http://www.reddit.com/								115
Slashdot	http://slashdot.org/								1728
Technorati	http://technorati.com/								1265

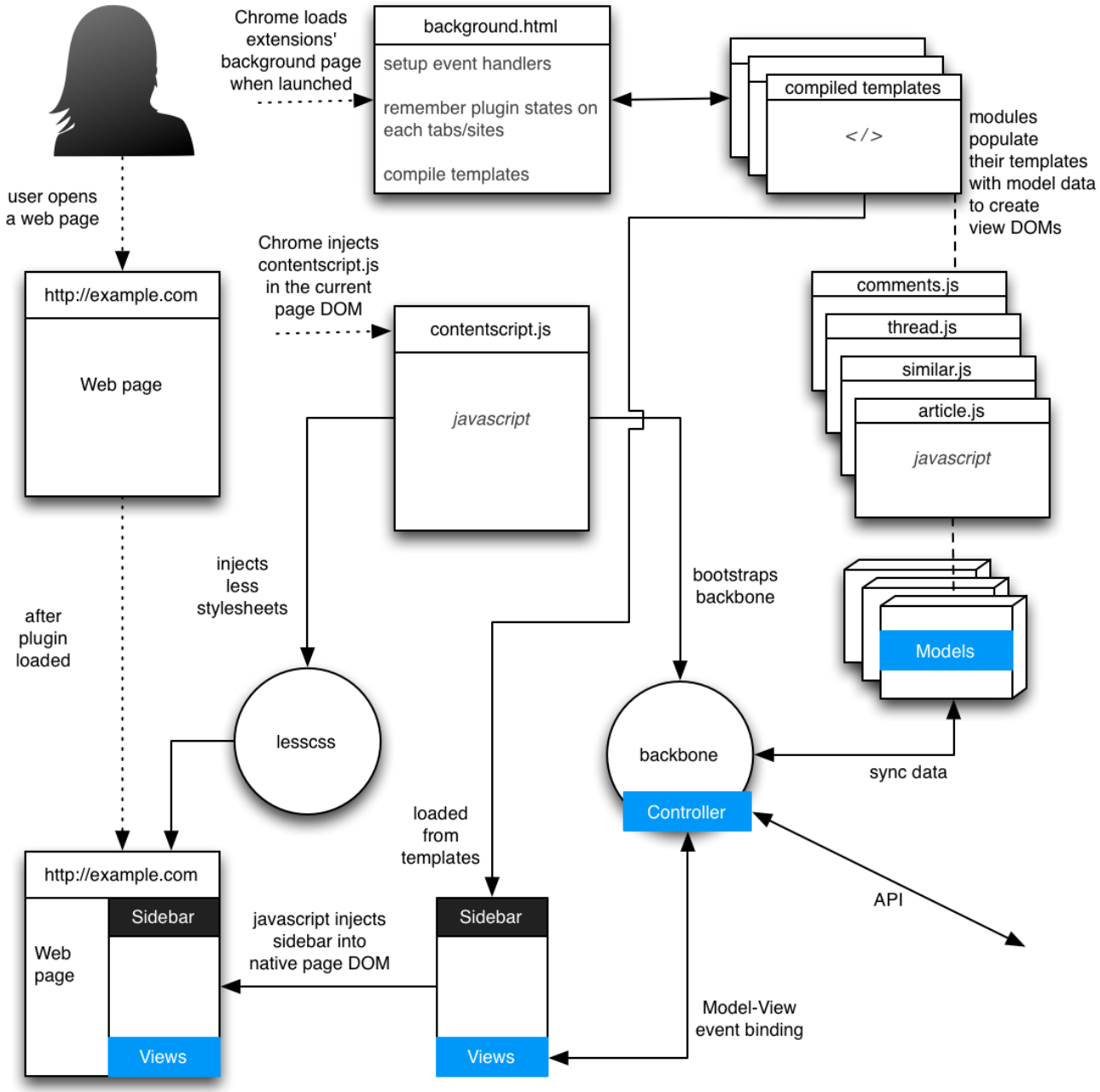
Backend Components Diagrams





Example: flow to response to API for related comments

Browser Extension Diagrams



Discovery Interviews

Interviewees

Evan M.
Hector A.
Grant G.
Jeannettte S.

Interviewee Profiles

Evan is a senior back-end programmer. He's in his mid-twenties and lives in Oakland.

Hector is a senior programmer, consultant, and small business owner. He's in his thirties and lives in Oakland.

Grant is a web designer, focusing on best practices and latest techniques. He's in his thirties, lives in San Francisco and works in the East Bay at a large company.

Jeannette is a master's student and high school girls volleyball coach. She is a teacher and works with young kids with special needs. She's in her late twenties, lives in San Francisco and works in the East Bay and Peninsula.

Interview Structure

These interviews were done in an exploratory manner. They began with questions about how the user browses the web. Then they focused on specific aspects of reading articles and comments. The goal was to get an idea of how people approach online discussions. During the interviews with Grant and Jeannette I also had the opportunity to watch them browse articles and comments. I performed a speak-aloud evaluation where they told me what they were doing as they browsed.

Interview Notes Synthesis

Evan

Evan uses the web to find answers to his coding questions. He reads Stack Overflow questions and discussions. He also reads Reddit and Slashdot. He is able to recognize people's Gravatar's (thumbnail image and username) from one website to the next. When he recognizes a Gravatar he has a prior notion of the person. So when they comment on Stack Overflow he'll have an idea of their reputation and knowledge before he even reads their comment.

Hector

Hector reads articles on the web, but he doesn't really read the comments. He will skim and sample the comments. If there is anything interesting he will read it in more depth. But predominantly he will read the articles instead of the comments. He uses the web to find answers to coding questions, but this is a different kind of browsing than reading articles for pleasure.

Grant

Grant uses his Google homepage to display preferred news and magazines. Each source has

its own grouping with a title and about four headlines. His interest is in design articles. There is a lot of opinions in design. When he reads an article he is interested in reading the comments to get a well rounded picture of the content. Specifically, he's interested in reading opposing opinions in the comments. If the article is biased one way he wants to read the opposing arguments.

Jeannette
Jeannette performs two types of browsing: pleasure reading and professional research.

For pleasure she reads articles on SFGate and multiple online magazines. She likes to use Flipboard as a content aggregator. When reading articles online she will start by looking at the top of the page, then scroll down to see the number of comments, then back up to read the article. The number of comments doesn't change whether or not she'll read the article. When she reads the comments she looks for entertainment in the form of drama. She likes to read longer comments, barely even reading or skimming shorter ones. She doesn't find value in short comments. She likes to read dialogue in the comments, especially when people are passionate about their opinion.

For professional research she reads scholarly magazines and academic research papers. Some articles are better in paper form so she might read Psychology Today after buying the paper form of the magazine. When reading academic research papers she will frequently read the references. When the references cite similar articles or authors she takes that into consideration. This suggests that they are more relevant.

Jeannette gave great insight into how she browses. She prefers to be in control of her browsing. Sometimes she starts with a Google search, and sometimes from a curated source. When she reads an article or comments she doesn't like to stray far from her original starting point. Sometimes she will read a name of a school or a technique and start a new search, or click on a link to read more about it. But she doesn't click on related articles or most other links. She described it as making her net too wide. She prefers to stay focused and begin a new search on her own instead of following links.

Interview Analysis
Our interviews suggest some things that we should consider as we develop our hypothesis and solutions.

From the interviews and analysis we are able to identify characteristics for initial user personas. We would need to do further interviews to validate, enhance, and alter these personas.

One way of classifying our personas is based on the question “How do people browse the web?” Based on this classification we identified three characteristics.

1. People who use aggregators such as the iGoogle homepage tools like Flipboard and Feedly. This involves building a list of self-curated subscribed sources which are displayed either chronologically (iGoogle) or through ranking and recommendation algorithms (Flipboard, Feedly).
2. People who have a few favorite sites they go to. They're exploring familiar sites like SFGate or Stack Overflow.
3. People who do Google search and explore. People with a question or query and they

start by searching.

Another way of classifying our personas is based on the question “How do people approach comments on the web?” Based on this classification we identified a second set of three characteristics.

1. They look at the reputation of the commenter. They identify reputation across domains by Gravatar, ID, Karma points, or reputation scores.
2. They don't care about who it is, they care about the dialogue.
3. They care about longer, well-articulated, well-reasoned comments.

Hypothesis

Our hypothesis is motivated by our belief that comments are valuable sources of diverse opinion and new knowledge. We observe and hypothesize the following.

People think comments are valuable.
Some people care about the reputation of the commenter.
Some people care about the dialogue surrounding the content.
Some people are looking for an opposing opinion.
People treat comments as different content from the source article.
Since people think comments are valuable, aggregating comments from the entire web (which are relevant to the current source article) is also valuable.

Findings From User Testing Grouped by Interview

Ariel H, 2012 Apr 9, 3:45-4:10pm

"Posts are only from Engadget?" **(Usability)**
Make clear exactly which sources are included **(Usability)**
Expecting newest comments first **(Consistency and Standards)**
Can't vote up/down? **(Interaction with the content)**
Assumes it's related articles **(What does this icon do)**
Assumes reputation or unique commenters **(What does this icon do)**
Scroll bar is invisible to the user. Also might consider putting it on the left side. **(Usability)**
Need more margin on the left and right. Give it some room. **(Design and layout)**
Make background more transparent **(Design and layout)**
Don't show "popular" filter in full text on every comment. **(Design and layout)**
Spell out the filter at the top. **(Usability)**
Don't need "at" and "on" in the comment info details **(Design and layout)**
Make comment clickable. Take the user to the target URL (the comment or article). And remove link text. **(Usability)**
Keep data visible in the comment without hover. **(Usability)**

Brendan C, 2012 Apr 9, 4:20-4:37pm

Comment should link to the comment itself **(Usability)**
He likes the comment info **(Content choice)**
What's determining "popular" status? **(Match between system and the real world)**
Is up/down vote part of our project or from the publication? **(Interaction with the content)**
Wants search functionality, not just using the browser's "find" feature (e.g. command+f) **(Feature request)**
Article filters - not sure about their meaning **(What does this icon do)**
Pick a word that is more familiar and applicable. **(Match between system and the real world)**
"Nobody reads text descriptions" **(Design and layout)**
Replies - is it comments that are replies to the article? **(Match between system and the real world)**
Hover effect (to view comment info) is good. Could consider design change instead. **(Design and layout)**
WK Note: When click on the user name show other comments by user on this site. **(Feature request)**
Only reads comments on Reddit and Stackoverflow. **(User Research)**
On Reddit expertise is available. Expertise is better than opinion. **(User Research)**
On Stack Overflow reads all the comments to see multiple ways of doing things. **(User Research)**
Doesn't read comments on sites like Engadget. (WK Note: It's not written down in my notes but I think the reason was because these sites have too much opinion rather than expertise.)**(User**

Research)
Search function would be more helpful than the filters we have **(Feature request)**
On Reddit the bubble up effect satisfies the filtering necessary. (WK Note: Make this automatic so the user sees only the bubbled up content.) **(User Research)**

Sebastian F, 2012 Apr 9, 4:51-5:13pm

"What are these letters?" **(What does this icon do)**
The hover effect doesn't fade the entire screen. It makes content pop and draws attention to it, rather than hiding it. **(Design and layout)**
"Are these comments all about the article?" **(Match between system and the real world)**
The browser plugin should tell the user where the comments are coming from **(Usability)**
WK Observed: The scrollbar is invisible. **(Usability)**
Link from comment should go to the actual comment in content (e.g. with an anchor to the comment in the comments section of the original source) **(Usability)**
Can I click on the up/down vote? **(Interaction with the content)**
WK Note: He wants to be able to up/down vote in our app. Assuming it works seamlessly. **(Feature request)**
What does "popular" mean? **(Match between system and the real world)**
Comments in threads with many comments doesn't mean it's good, only that there's attention on it. WK Note: His definition of popular means that it's good. **(Match between system and the real world)**
Deep - Show first comment from deep threads. **(Content choice)**
Hover effect (to see comment info) - The content will be there when I scroll anyway. It doesn't help enough. It could be implemented better. WK Note: He likes click better than hover for comment info. He likes descriptive text about filters. Show more info about results. **(Design and layout)**
Person Tab - wants to see more about the person. not just the comments by them. **(Content choice)**
Related articles - Great for berry picking for content. **(User Research)**
Add a troll filter to identify them. Give a clear indication of where trolls are commenting. **(Feature request)**

Elizabeth H, 2012 Apr 10, 10:26-10:57am

Normal browsing experience - Google for content, skim article, skim video, check out pics and charts (for comparison purposes), and quit reading an article if it's too long. She'd rather look for pictures. **(User Research)**
Regarding comments - skim for drama to crack up about, looking for gray (div) background (color) in the responses. Looking for snark. She's sad when she can't read "flagged" comments. Seeks out SFGate.com for humor/drama. It's a poorly written paper and audience writes crass things. **(User Research)**
Fading background is bad. It's distracting/confusing at first. **(Design and layout)**
Clicking on tabs must open the plugin window. **(Flexibility and ease of use)**

Showing “25 comments by 83 people” is confusing. It should be more like “25 comments from 3 publications”. Maybe hover over “publications” to see which are included. **(Content choice)**
Include title of the tab in the top (e.g. Related Articles). **(What does this icon do)**
Black on black scrollbar defeats the purpose. Can’t see it. **(Usability)**
Filters need labels. Didn’t know what they were. **(What does this icon do)**
People tab - Show profiles. Show number of comments on each publication. Show subject of actual comments and use filters. **(Feature request)**
Filters - Deep means profound, not the number of replies in the thread. Maybe use “Most replied to”. Maybe show filters in a dropdown and make “Most Popular” the default. Maybe use a dropdown list with “Sort by: ____”. What does “Long” mean? Use character count, or word count. **(Match between system and the real world)**
Hover for comment info - No opinion really. Not at first. Likes all at once to compare. Later she went back and said that she “does like the clarity now”. **(Design and layout)**
Comment info - Make it like Facebook or Twitter. Use a symbol and count. **(Content choice)**
Regarding up/down vote - show “Must be logged in” on click. **(Feature request)**
Comment date and time - Be sure to include just the date. **(Content choice)**
Put up/down vote above the date and location or on the side below the logo. Put more emphasis on up/down vote. **(Design and layout)**

Jeff Z, 2012 Apr 10, 2:55pm

Reads technology blogs through RSS readers on his iPad. He uses a Reader App that integrates with Google Reader. Also uses Flipboard sometimes. **(User Research)**
Almost never reads comments unless he thinks he might be interested in checking to see if comments are really stupid. He’ll also read comments on how to’s and product reviews. **(User Research)**
Normally assumes comments are not worth the time. He’ll skim 1 or 2. If there are hundreds of them it’s not worth the time to sort through them. **(User Research)**
Already reads a lot of content in RSS feeds. Doesn’t seek out “related content”. **(User Research)**
Normally he would read an article in full. He watches part of a video. It’s too long though (13 minutes). “Ooh graphs, how pretty.” Sometimes reads titles of Related Articles and may click. **(User Research)**
Not interested in trolling/flaming comments. One or two liners are like that. Likes longer comments. Long threads are full of crap comments. **(User Research)**
Likes fade out of the main page. May be distracting over time. **(Design and Layout)**
Left tabs - click and nothing happens. Tool tips are a little slow to be helpful. Not sure what the commenters tab would show. Maybe sorted by most comments but those people frequently have the worst comments. **(Match between system and the real world)**
The comment count in the left tabs feels nice. **(Content choice)**
Can see comments come from the site with the logo but not sure what article. **(Content choice)**
It would be nice to click on the article and have plugin hide. **(Feature request)**
Not sure how comments are sorted initially. **(Consistency and Standards)**
As for discoverability, click comment to read full comment is bad. Better to use ellipsis or “more”.

(Consistency and Standards)
Expects the link to go directly to the comment in the original article (using an anchor). **(Consistency and Standards)**

Comment info - doesn't want to see it all the time but motion on hover is distracting. Show comment info on click of a button for page. **(Design and layout)**
Up/down vote - doesn't really care unless on stack overflow. **(User Research)**
Filters - Popular, Deep - not obvious it would change the sorting or filtering. It's misunderstood. Understands they are different ways to sort. Doesn't know what the filters mean. Filters should all be labeled with nouns or adjectives, not both. **(Consistency and Standards)**
Related articles - expects other article titles related to the original article. List articles with better titles higher in the results. Comments are a better indication if titles are similar. **(Consistency and Standards)**

Evan S, 2012 Apr 11

Note about the test: Testing with the "new" top nav filters (full words in colored rectangles).
Hover effect with grayed out web content - Frightening. Wants everything in the frame. Needs to gray the whole page out entirely. Not helpful. Confused by it. **(Design and layout)**
He was clicking on top nav filters. **(Match between system and the real world)**
Double scrollbar - I don't feel like I'm in control. Might be the mouse. **(Usabililty)**
"So I'm on this article and these are comments related to this?" **(Match between system and the real world)**
Where are the "economic content pieces"? **(Feature request)**
Click on comment to read more of the comment wasn't obvious. **(Consistency and Standards)**
Comment info - language issue - Comments on threads is not clear. Depth of thread is a new concept. **(Match between system and the real world)**
"Where was I" just now? - He was on the comments tab but he had to find it again. **(User control and freedom)**
Are these articles related to this article, or this topic? Article text, and tags on the article. We need to play with this. "Why are you showing me these articles?" **(Match between system and the real world)**
Filters and sorting - Not clear the difference. **(Consistency and Standards)**
People - Wants to see comments from people in his social graph. Maybe use OAuth or Facebook Connect. **(Feature Request)**

Ryan M, 2012 Apr 17, 7:35-8:05pm

Word count - Clarify what it is. Quantifiable rather than qualitative (doesn't belong) **(Match between system and the real world)**
Top two button (tabs) pull up some content - confusing. **(Consistency and Standards)**
Other pull tabs are grayed out too much - didn't realize they did anything. **(Design and layout)**
Articles tab - Thumbnails aren't clear from logo and they caused fixation (distraction). Not clear user is looking at list of articles. **(Usability)**

Comments tab - Thinks of comments as from a user so show thumbnail of user not publication. **(Content choice)**
For an about page - use some sort of [i] icon (i for information) **(Content choice)**
Threading - display in a bigger window. **(Design and layout)**
Word count - long comments suggest ranting, or a lot to share, maybe an axe to grind, or that they're a domain expert. Long is just long. **(Content choice)**
Display "more" or a down arrow to see the rest of the comment. He didn't notice the ellipsis. **(Design and layout)**
Two finger scroll - Need indication of the end of the page. Not clear until the article scrolls down. **(Design and layout)**
Becomes familiar with user names on some sites. Wants to see their name with the comment. Uses username to identify people with an axe to grind. **(Content choice)**
Keep button for details (comment info) and "back" in one place. **(Design and layout)**
He recommended a slider interface he had seen recently for showing more or less comment info (e.g. comments, up/down votes, word counts). **(Feature Request)**
Possibly use shaded squares (like the Stock quotes app Walter uses) to indicate which page of detail the user is on. **(Design and layout)**

Findings From User Testing Grouped by Type

Usability

"Posts are only from Engadget?"
Make clear exactly which sources are included
Spell out the filter at the top.
Scroll bar is invisible to the user. Also might consider putting it on the left side.
Make comment clickable. Take the user to the target URL (the comment or article). And remove link text.
Keep data visible in the comment without hover.
Comment should link to the comment itself
The browser plugin should tell the user where the comments are coming from
WK Observed: The scrollbar is invisible.
Link from comment should go to the actual comment in content (e.g. with an anchor to the comment in the comments section of the original source)
Black on black scrollbar defeats the purpose. Can't see it.
Double scrollbar - I don't feel like I'm in control. Might be the mouse.
Articles tab - Thumbnails aren't clear from logo and they caused fixation (distraction). Not clear user is looking at list of articles.

Consistency and Standards

Expecting newest comments first
Not sure how comments are sorted initially.
As for discoverability, click comment to read full comment is bad. Better to use ellipsis or "more".
Expects the link to go directly to the comment in the original article (using an anchor).
Filters - Popular, Deep - not obvious it would change the sorting or filtering. It's misunderstood. Understands they are different ways to sort. Doesn't know what the filters mean. Filters should all be labeled with nouns or adjectives, not both.
Related articles - expects other article titles related to the original article. List articles with better titles higher in the results. Comments are a better indication if titles are similar.
Click on comment to read more of the comment wasn't obvious.
Filters and sorting - Not clear the difference.
Top two button (tabs) pull up some content - confusing.

Interaction with the content

Can't vote up/down?
Is up/down vote part of our project or from the publication? **(Interaction with the content)**
Can I click on the up/down vote? **(Interaction with the content)**

What does this icon do

Assumes it's related articles
Assumes reputation or unique commenters
Article filters - not sure about their meaning
"What are these letters?"
Include title of the tab in the top (e.g. Related Articles).
Filters need labels. Didn't know what they were.

Design and layout

Need more margin on the left and right. Give it some room.
Make background more transparent
Don't show "popular" filter in full text on every comment.
Don't need "at" and "on" in the comment info details
"Nobody reads text descriptions"
Hover effect (to view comment info) is good. Could consider design change instead.
The hover effect doesn't fade the entire screen. It makes content pop and draws attention to it, rather than hiding it.
Hover effect (to see comment info) - The content will be there when I scroll anyway. It doesn't help enough. It could be implemented better. WK Note: He likes click better than hover for comment info. He likes descriptive text about filters. Show more info about results.
Fading background is bad. It's distracting/confusing at first.
Hover for comment info - No opinion really. Not at first. Likes all at once to compare. Later she went back and said that she "does like the clarity now".
Put up/down vote above the date and location or on the side below the logo. Put more emphasis on up/down vote.
Likes fade out of the main page. May be distracting over time.
Comment info - doesn't want to see it all the time but motion on hover is distracting. Show comment info on click of a button for page.
Hover effect with grayed out web content - Frightening. Wants everything in the frame. Needs to gray the whole page out entirely. Not helpful. Confused by it.
Other pull tabs are grayed out too much - didn't realize they did anything.
Threading - display in a bigger window.
Display "more" or a down arrow to see the rest of the comment. He didn't notice the ellipsis.
Two finger scroll - Need indication of the end of the page. Not clear until the article scrolls down.
Keep button for details (comment info) and "back" in one place.
Possibly use shaded squares (like the Stock quotes app Walter uses) to indicate which page of detail the user is on.

Content choice

He likes the comment info
Deep - Show first comment from deep threads.
Person Tab - wants to see more about the person. not just the comments by them.
Showing "25 comments by 83 people" is confusing. It should be more like "25 comments from 3 publications". Maybe hover over "publications" to see which are included.
Comment info - Make it like Facebook or Twitter. Use a symbol and count.
Comment date and time - Be sure to include just the date.
The comment count in the left tabs feels nice.
Can see comments come from the site with the logo but not sure what article.
Comments tab - Thinks of comments as from a user so show thumbnail of user not publication.
For an about page - use some sort of [i] icon (i for information)
Word count - long comments suggest ranting, or a lot to share, maybe an axe to grind, or that they're a domain expert. Long is just long.
Becomes familiar with user names on some sites. Wants to see their name with the comment. Uses username to identify people with an axe to grind.

Match between system and the real world

What's determining "popular" status?
Pick a word that is more familiar and applicable.
Replies - is it comments that are replies to the article?
"Are these comments all about the article?"
What does "popular" mean?
Comments in threads with many comments doesn't mean it's good, only that there's attention on it. WK Note: His definition of popular means that it's good.
Filters - Deep means profound, not the number of replies in the thread. Maybe use "Most replied to". Maybe show filters in a dropdown and make "Most Popular" the default. Maybe use a dropdown list with "Sort by: ____". What does "Long" mean? Use character count, or word count.
Left tabs - click and nothing happens. Tool tips are a little slow to be helpful. Not sure what the commenters tab would show. Maybe sorted by most comments but those people frequently have the worst comments.

Feature Request

Wants search functionality, not just using the browser's "find" feature (e.g. command+f)
WK Note: When click on the user name show other comments by user on this site.
Search function would be more helpful than the filters we have
WK Note: He wants to be able to up/down vote in our app. Assuming it works seamlessly.
Add a troll filter to identify them. Give a clear indication of where trolls are commenting.
People tab - Show profiles. Show number of comments on each publication. Show subject of actual comments and use filters.
Regarding up/down vote - show "Must be logged in" on click.
It would be nice to click on the article and have plugin hide.
Where are the "economic content pieces"?
People - Wants to see comments from people in his social graph. Maybe use OAuth or Facebook Connect.
He recommended a slider interface he had seen recently for showing more or less comment info (e.g. comments, up/down votes, word counts).

User Research

Only reads comments on Reddit and Stackoverflow.
On Reddit expertise is available. Expertise is better than opinion.
On Stack Overflow reads all the comments to see multiple ways of doing things.
Doesn't read comments on sites like Engadget. (WK Note: It's not written down in my notes but I think the reason was because these sites have too much opinion rather than expertise.)(User Research)
On Reddit the bubble up effect satisfies the filtering necessary. (WK Note: Make this automatic so the user sees only the bubbled up content.)
Related articles - Great for berry picking for content.
Normal browsing experience - Google for content, skim article, skim video, check out pics and charts (for comparison purposes), and quit reading an article if it's too long. She'd rather look for pictures.
Regarding comments - skim for drama to crack up about, looking for gray (div) background (color) in the responses. Looking for snark. She's sad when she can't read "flagged" comments. Seeks out SFGate.com for humor/drama. It's a poorly written paper and audience writes crass

things.
Reads technology blogs through RSS readers on his iPad. He uses a Reader App that integrates with Google Reader. Also uses Flipboard sometimes.
Almost never reads comments unless he thinks he might be interested in checking to see if comments are really stupid. He'll also read comments on how to's and product reviews.
Normally assumes comments are not worth the time. He'll skim 1 or 2. If there are hundreds of them it's not worth the time to sort through them.
Already reads a lot of content in RSS feeds. Doesn't seek out "related content".
Normally he would read an article in full. He watches part of a video. It's too long though (13 minutes). "Ooh graphs, how pretty." Sometimes reads titles of Related Articles and may click.
Not interested in trolling/flaming comments. One or two liners are like that. Likes longer comments. Long threads are full of crap comments.
Up/down vote - doesn't really care unless on stack overflow.

Flexibility and ease of use
Clicking on tabs must open the plugin window.

Match between system and the real world
He was clicking on top nav filters.
"So I'm on this article and these are comments related to this?"
Comment info - language issue - Comments on threads is not clear. Depth of thread is a new concept.
Are these articles related to this article, or this topic? Article text, and tags on the article. We need to play with this. "Why are you showing me these articles?"
Word count - Clarify what it is. Quantifiable rather than qualitative (doesn't belong)

User control and freedom
"Where was I" just now? - He was on the comments tab but he had to find it again.