read.me

- PROPOSAL
- ABOUT
- TEAM

project read.me

Background:

I have been using RSS readers (Google reader, NetNewsWire...) for a long time, but none of them gave me a pleasant experience. The articles are always presented in a dull, linear fashion. I always end up seeing "1000+ items" in the unread count and feel like it will take forever to finish. I find Facebook newsfeed a great way of presenting immense amount of information: it picks the most interesting story from your friends and omit the less important ones. When we open the Facebook home page, we can look through important updates from friends and skip the meaningless information, such as an update from Farmville. I'm also inspired by various iPad RSS apps which present information in a newspaper / magazine way, which makes reading fluid and fun.

Goal:

Eliminate the need of tracking every single unread item on an RSS news reader. Instead, present the most useful information to the users. It will be a stream of high quality news and does not give users stress about reading all the unread items. Create a clean and intuitive user interface for users to read through articles.

Key Features:

Core Features

- Adaptive filtering shows articles you are interested in and filters out the irrelevant ones
  • Implicit and explicit user feedback tracking
  • UI support for user feedback
- User friendly interface
  • Always show the latest articles, as opposed to unread articles
  • UI follows newspaper and magazine layouts

Extra Features

- Find and subscribe RSS feed by one click on the bookmarklet
- Use your Twitter account shares as training data
- Repeat article detections to avoid errors in handling repeat articles
- Automatically generate tags for articles. user can search for articles based on tags
- Trends
- Track changes in user’s interests

Implementation:

To make recommendation engine work, here are some ideas

- Naive-Bayes classifier algorithm
- Track your twitter account and google reader account, find shares you have made. It will say what topics you're most interested in
- You can define keywords / tags that you want to follow
- Find similar users and recommend their favorites
- Crawl through popular news sites and blogs and find the tags of their articles, and find how many times an article had been retweet, Facebook like, buzz, or dig

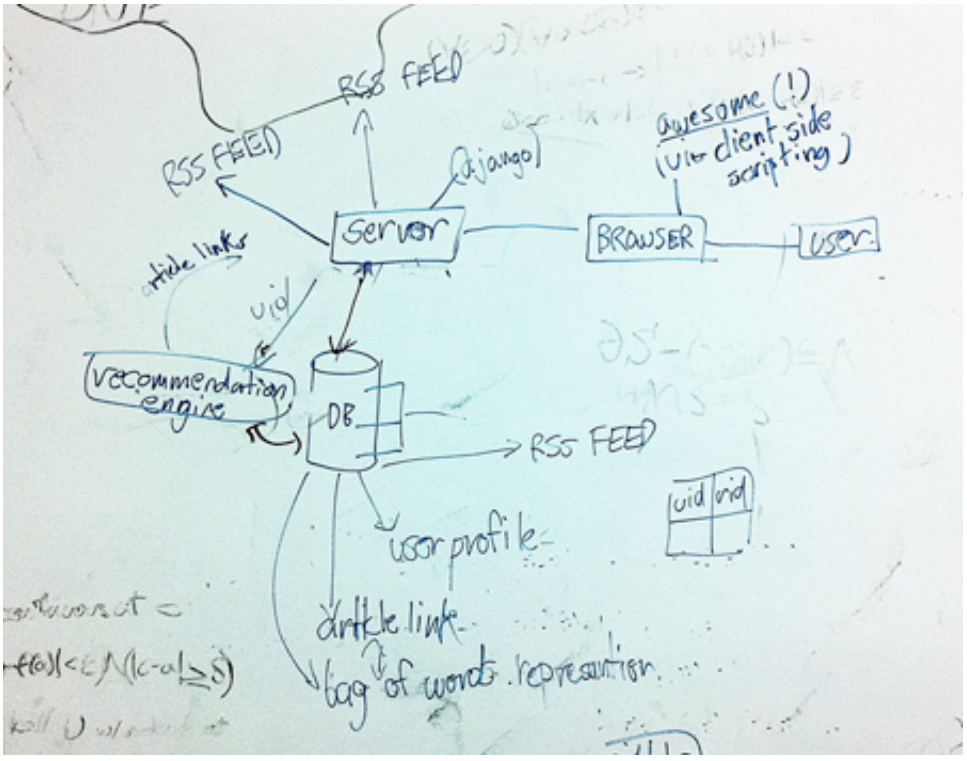
UI design inspirations:

Below are some websites or applications which I got inspirations

- The New York Times Skimmer
- AP Timeline Reader
- Flipboard for iPad
- Times for iPad

Since this application is on the web, we will design it for computer screens and mouse/trackpad use.

Application Architecture:



- RSS reader UI design and implementation by HTML, CSS and Javascript
- Use Django (Python) as web development framework for back end development

- Database that stores user preferences and article information (bag of words) in MySQL
- RSS and web crawler (dependent on RSS reader UI)
- Recommendation Engine (Recomengine) possibly implemented by Python or Java. Communicate with Django by Python function calls or through web service (dependent on Database)

Team

- Sean Ren - UI Design, front end dev, Django dev, Recommengine
- Peter Scheibel - Database (models) implementation, Recommengine
- Kha Nguyen - Django dev, Logo design, Recommengine
- Michael Mathews - Front end dev, Recommengine

Schedule and Milestones:

- Oct 29 - milestone 1 due
  • RSS reader UI basic layout and function, should display all feeds
  • Research on recommendation engine and it should have some sort of algorithm by this time
- Nov 17 - milestone 2 due
  • UI enhancements
  • Recommendation engine should be significantly improved and integrated with the UI
  • Should collect fair amount of user data
- Dec 3 - code complete; start experiments and writing
  • System polished and recommendation engine fine tuned
  • Test for volunteer users
- Dec 14
  • Presentation, reports, and code turn in

Machine Learning:

To determine the relevance of a particular article for a particular user, we are planning to use a classifier (possibly naive bayes with incremental updates). We are planning to gather data by releasing read.me to the web and/or friends and let them use the service for a period. We will track users' behaviors, such as mouse clicks, favorites, and shares. Also, we can learn about people's preferences from our test users' twitter, buzz, or google reader account. Popular websites and blogs is another probable location we collect data, where we can find tags and popularity of articles.

Evaluation:

The system can automatically determine the recall and precision of the algorithm by comparing it's evaluations with the user's evaluations (so, the effectiveness of the classifier could be determined simply by having people use the system). User tests could be conducted in two ways:

1) Have each test user define their own profile

and/or

2) Define user profiles and have test users evaluate relevance of articles based on the profiles. This would

make it easier to accumulate more time in a user profile (even if each test user spends a small amount of time on the profile, they are all contributing to the same profile). However, this approach may be subject to error due to individual bias.

To evaluate the UI, we can have test users complete predefined use cases and track issues like periods of confusion, or having to ask how to proceed.