CASE WESTERN RESERVE
UNIVERSITY EST. 1826

Project October

Read news that you want to read

Authors:	Rajesh Cherukuri, Tom Dooner, Mika Little, Brian Stack
Project:	Project October
Date:	February 26, 2013

Contents

List of Figures

1 Abstract

In modern news aggregation services, such as Reddit, Slashdot, Digg, and Hacker News, longtime members report noticing a marked decrease in quality of discourse as the services gain mainstream attention. This gradual but irreversible decline caused by the influx of new users dates back to the newsgroup era, when new college students would log in for the first times in September. After AOL opened newsgroups to the masses, the community standards continued to devolve, according to newsgroup veterans[**?**]. This effect came to be known as "Eternal September".

The inevitable loss of the longtime members further perpetuates the problem, and causes large numbers of excellent contributors to feel out-of-place in their own community. We aim to engineer a service that uses technological principles to avoid this, thus improving the user experience and allowing a large community to benefit from thoughtful discourse and interesting articles.

2 Project October: Hybrid Recommender System

Due to the enormous amount of information available online, the need for a highly developed personalization and filtering system is growing permanently. Recommender systems constitute a specific type of information filtering that attempt to present items according the interests expressed by a user[**?**]. Most web recommenders are employed for e-commerce applications or customer adapted websites, which assist users in decision making by providing personalized information [**?**], but the same techniques that suggest related items on e-commerce websites can recommend news articles to users who will enjoy reading them. We believe Project October is the first attempt to apply recommendation techniques to social news aggregation.

2.1 Background

It is our hypothesis that recommendation of news sources will provide a scalable community experience that can be tailored to each person's interests. Providing an automatic, customized, recommendation of articles will prevent the community from being diluted by new users and thus prevent the Eternal September phenomenon. Users that prefer intellectual discourse will automatically be recommended articles and comments that fit such a criteria, and users that prefer simpler entertainment will be recommended such stories.

This proposal discusses an implementation of a hybrid collaborative and content-based filtering approach for a web-based recommender system ("October"). In particular, we will be linking various news sources and user submitted sources. The resulting network of user-item relations and associated content features is converted into a unified mathematical model, which is applicable to our underlying neighbor-based prediction algorithm. By means of various experiments, we demonstrate the influence of supplementary users as well as item features on the prediction accuracy of October, our proposed hybrid recommender. In order to decrease system runtime and to reveal latent user and item relations, we factorize our hybrid model via singular value decomposition (SVD).

For the development and evaluation of our proposed hybrid recommender system, we make use of various news outlets and user recommendations, importing them into a graph database. Both corpora are joined in a unified mathematical model, which describes the complex network of interdependencies.

2.2 Intended Audience

We intend Project October to be open to the public. As such, we expect the user base to be somewhat technical in nature at first, some of the Internet's power users.

2.3 Application

2.3.1 Frontend

The frontend will be a standard social news aggregator. Upon going to the homepage, logged-out users will be presented with a splash page that describes the allure of Project October and offers a registration link. Users can register for the application by selecting a username and password.

When logged in, users will be presented with their personalized content. We are modeling the design after the front page of a newspaper – articles that we judge to be more interesting to a user will be placed in the most prominent positioning, and articles that are less important will fill the side columns and occupy less space. Users will be able to click on the headline (or associated image, if existent) to be taken to the original news source.

Each news article will feature a link to view the associated comments and it will feature a corner icon of how much credibility ("cred") a given article possesses. This is analogous to upvoting in Reddit, except that giving an article cred on October will inform the recommendation engine of your individual preference, rather than directly impacting the weighting of an article by a predefined formula.

2.3.2 Backend

The backend will contain a recommender system, which will consist of a text classifier for incoming & existing articles to be indexed into a graph database, a learning algorithm to combine what the recommender has learned about the user with the index information, and finally an API to receive queries and return results. The text classifier will be a combination of Latent Semantic Analysis with Singular Value Decomposition and Latent Dirchlet Allocation [**?**], allowing for the clustering of learned topics to create categories of topics while maintaining coherence to produce an index similar to that formed by human judgement. As for the learning algorithm, we will be using a custom bayesian framework [**?**] that will allow for weighting based on a particular user's current interest instead of a stored calculated interest or utilizing the general population's interests. Finally, the method of receiving queries and returning results will utilize an API, which is discussed in the following section.

Essentially the approach works as follows:

1. October predicts the user's authentic news interests regardless of the news trend, using the user's clicks in each past time period

2. Predictions made with data in a series of past time periods are combined to gain an accurate prediction of the user's authentic news interests

3. October predicts the user's current interests by combining their authentic news interests and the current news trend in the cluster the user belongs to within the property graph.

2.3.3 Frontend/Backend API

October will employ an API to promote separation between the frontend and the backend recommender system. This will provide a clear interface and facilitate easy simultaneous development of both parts of October.

To implement the API, we will use Apache Thrift, an Interface Description Language[**?**]. The essence of the API is simple, featuring primarily two types of calls:

Give recommendations for user n This is the main output from the backend, returning recommended news stories or comments for a given user. Ancillary parameters will be added to this to facilitate the frontend placement of articles, e.g. the recommendation confidence and individual article weightings.

User took action n This is the main input to the backend, allowing it to adjust recommendations according to user action. The parameters to this API call can be of many types. For example, "User commented on article $\#n$", "User gave cred to comment $\#n$", and "User visited link $\#n$" are all valid parameters for this API call.

These two calls manifest themselves in Thrift with a few simple object definitions and an interface. The abbreviated version `0.1.0` is attached as Appendix **??**.

2.4 Project Management

We will manage the project using agile development methodologies with a focus on progressive elaboration and documentation for the purposes of this project.

2.4.1 Agile Development

In order to stay within the bounds of Agile Development while still maintaining the requisite documentation, we will be creating documentation at the end of each release. In order to schedule work, we have employed Pivotal Tracker, an online task management solution. From the queue, we will assign tasks as we finish previous tasks. The work naturally fits into two main categories – frontend and backend. Work relating to each category will be assigned to the same people until we make enough progress to cross between teams at will. At least until that point, Brian and Rajesh will work on the backend while Tom and Mika create the frontend.

2.4.2 Scrum

We will meet every Monday, Wednesday, and Friday at 3:55pm to have a short meeting to discuss the current progress, any stumbling blocks, and what work is coming up in the immediate future.

2.4.3 Release Schedule

Releases are bi-weekly and will include a team review as well as planning for the next release. Release dates will involve pushing the current state of the project to a continuous integration service (http://travis-ci.org), after which automated deployment will be executed.

2.4.4 Iterations

We plan to have weekly iterations. At the end of each iteration, team members will re-sync and make sure that we have a clear outline for the next iteration.

2.5 User Interface

2.6 Testing & Evaluation

2.7 Project Progress

2.7.1 Discussions & Conclusions

3 Appendix

References

[1] Gediminas Adomavicius , Alexander Tuzhilin, *Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions*, IEEE Transactions on Knowledge and Data Engineering, v.17 n.6, p.734-749, June 2005 [doi¿10.1109/TKDE.2005.99]

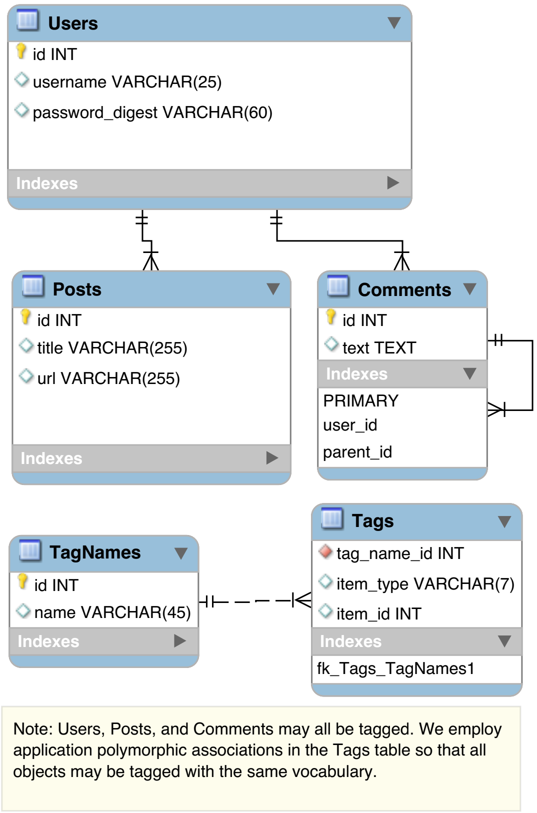
[2] Greg Linden , Brent Smith , Jeremy York, *Amazon.com Recommendations: Item-to-Item Collaborative Filtering*, IEEE Internet Computing, v.7 n.1, p.76-80, January 2003 [doi¿10.1109/MIC.2003.1167344]

[3] Jensen, V. *Bayesian Networks and Decision Graphs*. Springer, 2001

[4] Arthur Asuncion, Padhraic Smyth, Max Welling. *Asynchronous Distributed Learning of Topic Models*. NIPS 2008.

[5] `http://en.wikipedia.org/wiki/Eternal_September`

[6] `http://en.wikipedia.org/wiki/Apache_Thrift`



Note: Users, Posts, and Comments may all be tagged. We employ application polymorphic associations in the Tags table so that all objects may be tagged with the same vocabulary.

Figure 1: Frontend Database Relational Diagram

A Thrift API Definition

```
#
# Structs
#

/** A single post with its calculated weight.
 * @param post_id, the unique id of a post.
 * @param weight, the "importance" of the post to the querying user [0,1].
 */
struct Post {
    1: required i64 post_id,
    2: optional double weight,
}

/** A list of posts along with a confidence for the accuracy of the list.
 * @param confidence, the confidence in the results.
 * @param posts, a list of Posts.
 */
struct PostList {
    1: optional double confidence,
    2: required list<Post> posts,
}

/** The types of actions that can be performed in a triple (subject, verb, object) */
enum Action {
    READ,
    VOTE_UP,
    VOTE_DOWN,
    VOTE_NEGATE,
    POST,
    COMMENT,
    REPORT,
    TAG
}

#
# Exceptions
#

/** The queried object does not exist. */
exception NotFoundException {
}

/** There was an error processing the request */
exception EngineException {
    1: required string why,
}

/** Request took too long to process */
exception TimeoutException {
}

#
# Service
#

service Recommender {

    /** Test for connectivity */
    string ping() throws (1: TimeoutException te),

    /** Request a list of posts that are most appropriate for a user
     * @param user_id, the user that the posts are being requested for
     */
    PostList recPosts(1: required i64 user_id) throws (1: NotFoundException nfe, 2: EngineExcepti

    /** Alert the recommender that a user has actioned a post
     * @param user_id, the user that performed the action
     * @param verb, the action taken (this is from the Action enum)
     * @param post_id, the post that the action is being performed on
     */
    void user_v_post(1: required i64 user_id, 2: required Action verb, 3: required i64 post_id) t

    /** Alert the recommender that a user has actioned a comment
     * @param user_id, the user that performed the action
     * @param verb, the action taken (this is from the Action enum)
     * @param comment_id, the comment that the action is being performed on
     */
    void user_v_comment(1: required i64 user_id, 2: required Action verb, 3: required i64 comment
}
```

B Database Design

B.1 Frontend Database

B.2 Backend Database

C User Manual

D Programmers Manual