Project: Open Access Publishing Service

Due: tentatively 31 December 2018

1 Background

In his 2012 essay "A World Without Referees", Larry Wasserman argues that the current academic publishing system, peer review, is outdated (as it was invented 350 years ago) and bad (as it makes dissemination of new knowledge hard). There should be a modern, non-blocking way to spread new scientific knowledge as widely as possible. He suggests creating "a marketplace of ideas" where everyone is free to display his ideas, the quality of which is determined not by a few arbitrarily chosen experts, but by many and most importantly, by time. He describes the arXiv model and the "publish on your own website" model as two alternatives to the restrictive, closed science model.

The current publishing is also expensive and the cost has been rapidly increasing, which further impedes spread of new knowledge. University libraries, even the richest ones like Harvard Library, are crippled by high journal subscription fees. To counter such paywalls, the Harvard University has initiated the Harvard Open Access Project. More recently, European universities and research organizations asked their employees to publish their work only in journals that offer immediate open access, a move called Plan-S.

2 Project description

This project concerns free, community-based, and open academic publishing. The non-technical and technical aspects of this project are described below.

2.1 Non-technical

- Provide open access publishing service for the Zhejiang Normal University, extensible to other universities in the world. Therefore, we should use UTF8 as our encoding method.
- The authors using the service are free to create new subjects (forums). However, no duplicate subjects are allowed.
- The authors using the service are free to contribute new articles. We assume that they will be responsible for the quality of their own work.
- The authors using the service are not required to register for publishing. Instead, they just need to provide their valid email addresses. In the rare occasions where someone continually pollute the service with rubbish articles, the service administrator can blacklist his email address for a period of time.
- The authors using the service are charged with no fees.
- The articles are freely downloadable as PDF files. Each article page includes article title, publication date, authors' contact information, an abstract, and a Highlights section describing what the exciting things are in the article.
- Adopt a donation approach as the Mozilla Foundation. Perhaps a donation page containing payment methods is useful.
- Adopt a Commit-Then-Review model, which means articles will be published immediately under a selected subject after the author(s) clicked the submit button.
- The articles published to the service can be commented by unlimited number of reviewers, not restricted to a small group of people selected by a journal editorial board. Therefore, the service can be considered as a more democratic peer-review model. For simplicity, only allow text comments. These comments are shown in the article page and are worldwide visible. Discussions should center around the materials presented in the article and therefore personal attacks or other irrelevant comments are prohibited. Think about a way to ensure this.

- To post, the contributor must at least provide a valid email address. Think about how to prevent email address harvester from maliciously spamming the contributors.
- Each article, as well as each comment, can be up-voted or down-voted. Think about how to prevent duplicate voting.
- Each article has a *popularity metric* taking into account the following information: number of visits, number of comments, up-votes, and down-votes. Define an appropriate metric.
- By default, articles in each subject page are shown in reverse chronological order. In addition, the articles with highest *popularity metrics* will be displayed in a prominent place for a limited period of time.
- Each subject has a subject page displaying its articles. Each article has an article page displaying its relevant information and a link to download the full article. Each author has an author page displaying the articles he has published along with their popularity metrics.
- Provide a convenient way to search for articles, comments, or authors, as Microsoft's Bing does.

2.2 Technical

- Identify objects, actions and define classes. For example, we can consider these classes: `Post` (with subclasses `Article` and `Comment`), `People` (with a subclass `User`), `Database` (with a subclass `Publisher`), and `Subject`, among others. Define all classes in a single source file, `service.py`, which can be imported as a module.
- Use SQLite or other light-weight database management tools. Minimize the number of tables to use.
- Use Flask or other web frameworks. Design well-organized URLs, for example, as follows:
  - `/post`
  - `/subject/subjectID/post`
  - `/article/articleID`
  - `/author/authorID`
  - `/subject/subjectID`
  - `/subjects`
  - `/comment/articleID`
  - `/upvote/articleID`
  - `/downvote/articleID`
  - `/upvote/commentID`
  - `/downvote/commentID`
  - `/search`

  It might be reasonable to use email addresses as authorID. How to generate article IDs? How to generate comment IDs?

2.3 Users

Visitors, article contributors, comment contributors, and service maintainers. The service maintainer has direct read/write access to the database. The article contributor and comment contributor must provide valid email addresses to post an article and post a comment, respectively.

2.4 Other features

- Plain HTML pages without any heavy flashes or JavaScripts.
- Respond any requests in less than 5 seconds.

3 Requirements

- Do the project in a group with a maximum of 2 students.
- You will present your progress a few times in this semester (the presentation schedule to be announced). We will produce a Progress Monitoring Table for each group.
- I only accept object-oriented implementation in `service.py`.
- Keep a commit history of your work using git. Use the online git repository `https://www.gitee.com`. Use your real names when you sign up for gitee. Create a **private** project on gitee so that other students cannot see your work.
- Submit by the due date use cases and class diagrams.
- Submit by the due date working python source files.
- Submit by the due date a software demo (a screencast for example).

4 Frequently Asked Questions

TBD.

This document might undergo a few small changes in the future. I will let you know if that happens. In the meantime, if you have any questions, don't hesitate to ask me via email or in person.