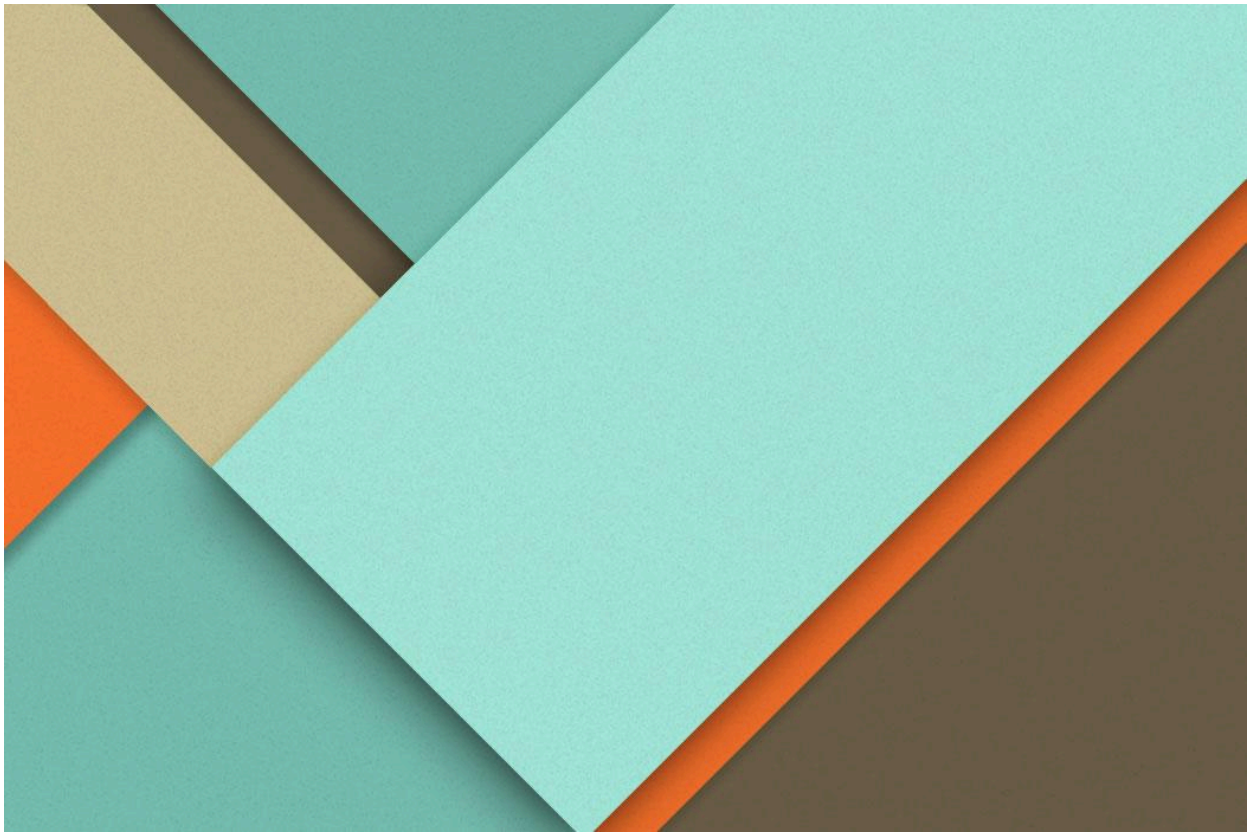

Accuracy Review of Wikipedia - GSoC 2016 Proposal

Wikimedia Foundation

Phabricator task containing the proposal: T129536

15.03.2016

—

Priyanka Mandikal
BITS Pilani K.K. Birla Goa Campus, India

Name and Contact Information

Name: Priyanka Mandikal

Email: priyanka.mp2808@gmail.com

IRC nick: prnk28

Blog: priyankamandikal.wordpress.com

GitHub: priyankamandikal

Location: Goa, India

Timezone: Kolkata, INDIA, UTC+5:30

Typical working hours: 6 – 8 hours from 10 am to 12 midnight (local time)

Synopsis

This project aims to provide editors with an easy mechanism to find and review inaccurate content in Wikipedia articles. The aim is to achieve this with the help of algorithms that are specifically designed to flag suspected content, which the editors can then review by a three-way mechanism. The reviewer bot finds articles in given categories, category trees, and lists. For each such article, the bot aims to accomplish the following tasks:

1. Flag passages with facts and statistics which are likely to have become out of date and have not been updated in a given number of years
2. Flag phrases which are likely unclear
3. Flag bad student edits

Flagged instances are then stored in a database and open to the reviewers for reviewing and closing the flags. A reviewer can decide to either 'Accept' or 'Dismiss' a review item as valid/invalid. Two reviewers decide the fate of each review item, and in the case of a conflict, it is sent to a third reviewer. Reviewer reputation scores are computed based on a mechanism of acceptability of reviews by other peer reviewers. Reviews which lowered the scores can be optionally displayed to the reviewers.

The interface is being implemented in Python Flask and uses SQLAlchemy ORM for the database.

What it means to accomplish

Provide a mechanism for auto-flagging specific content to be reviewed and present them to the reviewers.

Step 1: Develop a login and authentication system for the reviewers

Step 2: Design the review item queue database

Step 3: Integrate the workflow as depicted here

Step 4: Develop and implement algorithms to help in each of the aforementioned tasks

Note that Step 1 has already been implemented as part of the micro tasks during the contribution stage.

How will it benefit Wikimedia objectives such as these?

- It will assist the editor community in drawing attention to specific parts of an article that need to be reviewed for copy-editing, rather than the existing process of just flagging entire articles
- It'll provide a robust mechanism for review items to be analyzed by different reviewers
- It will help in realizing the broader content reviewal objectives of the Wikimedia community

Mentors

- Primary mentor - James Salsman
- Co-mentor - Fabian Flock
-

Deliverables

The main deliverable will be a working web application for editors to review flagged content, which explicitly pinpoints inaccuracies within an article.

Required Deliverables

Phase I

-- Design and model functions to implement the flagging of content (**milestone 1**)

Phase II

-- Design the review item queue database (**milestone 2**)

-- Implement the manual list based input functionality (**milestone 3**)

Phase III

-- Implement the reviewer reputation scoring system (**milestone 4**)

-- Add accepted flags to a published list on a webpage (**milestone 5**)

- Testing and Documentation.

Schedule

Community Bonding Period (May 10 - May 22)

- Remain in constant touch with my mentor(s) and community.

- Explore features that can be used for flagging content, by actively editing wiki articles. This would, for example, help me in better understanding diffs, permalinks, ordinary article URLs, etc.

Obtaining article dumps (1 week, May 23 – May 29)

-- Obtain article dumps and device efficient mechanisms to iterate over them so as to run flagging algorithms on each one

-- Discuss with mentors on how Wikiwho can be incorporated into the project at this stage for obtaining authorship (for student edits), revision dates (for outdated content), etc

Implement one flagging algorithm (2 weeks, May 30 – June 12) - __Milestone-1__

- Discuss with mentors on devising algorithms for flagging outdated content from the retrieved dumps

-- This could involve mechanisms to detect outdated weather/ climate, population statistics on geographical content and so on based on revision dates

- Also include the previously implemented readability tests into the automatic flagging system

- Continue working on the algorithms throughout the internship process

Review item queue database design and creation(1 week, June 13 – June 19) - __Milestone-2__

- Begin Phase II of the project

- Based on what I have inferred via editing articles and upon discussion with my mentor, set up the review item queue database.

- This would include review id, article name, wiki type, link, link type (diff, permalink, url, etc), pertinent excerpt text, source of review item (manual entry, student editor, poor readability scores, outdated content, etc) and so on

- Research into the relationships between databases in SQLAlchemy ORM and define relationships between the Reviewer and the Review item queue databases.

Mid-term evaluation

- Submission of completed code

Manual list based input (2 weeks, June 20 - July 3) - __Milestone-3__

- Create a reviewer profile page

- Create a webform with which a reviewer can manually enter an item that needs to be reviewed.

- Implement methods to add these items to the review item queue database created earlier.

-- Methods for fetching the form content and updating the database

-- Methods for querying and displaying this database

- Write testcases for exhaustively testing the implemented methods

Research Period (1 week, July 4 - July 10)

- This period will be in preparation for executing Phase III of the project

- Design the architecture of the reviewer reputation scoring system. It should handle the following two cases

-- Scenario 1: Both the reviewers agree on the decision of accepting or dismissing the flag

-- Scenario 2: Upon disagreement, send this item to a third reviewer for consensus

- Also implement ways to send the automatically flagged content into the queue

Integrate with the workflow (1.5 weeks, July 11 - July 20)

- Every item on the queue must be served to at least two reviewers

-- Randomly allocate every item to two reviewers

-- Keep track of the status of the item and serve it to a third reviewer if necessary

Implement the reviewer reputation scoring system (2 weeks, July 21 – Aug 3) - __Milestone-4__

- Write methods for computing the reputation scores of reviewers

- Use a sound scoring method that uses the percentage of responses that were agreed upon by peer reviewers as a metric

-- Ability to display responses that reduced the reviewer's score

-- Add options to view such responses on the reviewer's home page

- Testing of the written scoring functions

Publish the list of accepted flags (1.5 weeks, Aug 4 – Aug 14) - __Milestone-5__

- Every flag that is accepted as valid by the reviewers are to be continually published on a web page and removed from the review item queue database. Flags that have been dismissed must also be removed from the db.

- The published list is a chronological list with the ability to view 100 items at a time

-- Discuss with mentors about the possible ways of handling this task

Wrap up (2 weeks, Aug 15 – Aug 23)

- Wrap-up phase

- Edit documentation and code

- Testing

- Community Review

- **Submission to Google**

About Me

Education

University: BITS Pilani Goa Campus, India

Major: Computer Science (B.E.) and Physics (Int. MSc.)

Degree level: 4th year undergrad

Graduation year: 2017

Why this project is important to me

As I was working on the micro tasks for this project, I got more interested in how Wikipedia deals with outdated content and if there is a mechanism to detect inaccuracies in facts, if any. Although it has working systems for detecting vandalism (ClueBot_NG) and quality of revisions (Huggle), I learnt that there is no explicit mechanism for checking facts and reviewing the accuracy of content that hasn't been updated in a while. In addition, many bots that are responsible for flagging poorly structured articles tend to flag entire articles (!) rather than the culprit sentence constructs/passages. So copy-editing long articles is a really painful task!

This motivated me to pursue this project right from December onwards when my winter break began. I actually spent quite some time researching on the application's architecture, use cases, database options and flagging mechanisms before coding. This was very fruitful

in the sense that as I progressed in the coding phase, it made so much more sense to have spent ample time on the design phase, which I think is crucial for any novel project. I completed the login system for reviewers in the time I was working on the project and also implemented a couple of readability tests that spew out scores for passages/articles based on their level of 'understandability' for the reader.

Another major motivator for pursuing this project is my interest in the areas of Natural Language Processing and Artificial Intelligence. In the current internship timeline, I will be starting off by implementing a few basic flagging algorithms and then proceed to integrate them with the workflow. The aim is to keep testing and iterating over different techniques throughout the internship period, to find out what works and what doesn't as well. I am excited to know how this project might pan out once strong flagging algorithms are brought into the picture!

This project is important to me, not only as an internship experience, but as a learning process itself. I hope to give and take back a lot through this project.

Past Experience

Contributions to this task

I have been actively working on this project since mid-December. I was constantly in touch with my mentors and spent quite some time envisioning the architecture of the entire application before beginning to code. During the coding phase, I went over various tutorials on building web applications in Python Flask and familiarized myself with the tools needed. I then went ahead and built the entire login system for reviewers along with exhaustive test cases to check the authentication and secure password-hashing functionalities.

I have provided links for the project repository and blog below.

Project GitHub Repository

Detailed Documentation

I have written over a 1000 lines of code. The detailed documentation for this can be viewed in the project blog

In brief, I implemented the following features as part of the login system:

- Basic app structure in Flask with templates for base, index, error pages, etc. This also involved setting up the configuration environments file for development, testing and production.

- Created the reviewer database in SQLite using SQLAlchemy ORM. The attributes are id, email, username, password_hash, confirmed, agreement, and reputation.

- Password Hashing and Authentication Blueprint creation – used the Werkzeug package for hashing passwords so as to securely store them in the database. Test cases were also implemented.

- Implemented user authentication with flask-login package for handling reviewer logins. Also made use of the UserMixin class.

- Added a registration form for new users which has methods for validating email and username from the already existing values in the database.

- Implemented confirmation mail sending functionality that sends a confirmation link to the specified email id during registration and activates the account once it has been validated. Made use of 'itsdangerous' package which provides cryptographically signed cookies for generating confirmation tokens.

- Functionality for Updating and Resetting passwords was also implemented

Note that all the above code has been extensively tested and can be verified in the code base.

Apart from the login system, I worked on implementing a couple of readability tests such as SMOG index and Flesch-Kincaid Readability test, which score texts based on the level of readers who can understand the text. The idea behind the scoring system is based on the number of syllables per word, number of words per sentence, etc. The documentation and code for the same can be viewed below.

Readability Tests

Other Contributions

I actually began exploring Open Source organizations in November 2015. Prior to that, I had focused on working on personal projects or in small groups. Exploring the FOSS community has really opened my eyes to the learning opportunities that working in larger communities brings to the table. The exchange of ideas and mentorship that happens has truly been inspiring in more ways than one.

Under the 'personal projects' category, I have worked on a couple of machine learning projects and have gained experience in dealing with large data sets, feature extraction, as well as training and testing with various ML techniques such as classification, regression, clustering, Adaboost, SVMs, etc. I am currently learning about neural nets and hope to complete two major projects in deep learning and artificial intelligence by the end of this semester. Here's to hoping this knowledge can come to use in this project itself! I will use the time before the official internship starts for researching into different approaches that are currently being employed by bots, and start conceptualizing our own approach for this specific task at hand. We could probably reach out to the wiki-ai community for suggestions and ideas.

Details about specific projects

- Prediction of protein-protein interactions using ML techniques - Have been involved in a bioinformatics project at college. It involves the use of machine learning techniques to predict protein-protein interactions. Coded in Python and made use of the ML libraries available in scikit-learn. This project has exposed me to a variety of ML algorithms and feature generation approaches.

- CTR Prediction - Worked on a data science project to predict the advertisement click-through rate. Implemented the C4.5 Decision Tree algorithm in Java using the Hadoop framework. Had a hands-on experience on dealing with big data and data distribution.

- Mozilla Feedback Analysis - Built a data scraper and explorer to analyze feedback data from Mozilla's feedback page. Learnt quite a bit about preprocessing and visualizing data and deriving meaningful insights from it.

- Resource Management System - Built a GUI based application in Java for booking rooms and cabs in the institute as part of a course project in Object Oriented Programming. This project laid a solid foundation in building applications from scratch, right from the requirements and design phase, to the coding and testing phases.

- Currently following an online Deep Learning course (CS231n) offered by Stanford. Will complete a major project in the same as part of a semester project. This experience should help me implement a neural net, if there is scope for one during the internship.