Summer Fellowship Report

On

FOSSEE Forums Web Development

Submitted by

Chaitanya Baranwal

Under the guidance of

Prof. Kannan M. Moudgalya
Chemical Engineering Department
Indian Institute of Technology, Bombay

July 4, 2018

Acknowledgment

First and foremost, I would like to thank Prof. Kannan Moudgalya for establishing this fellowship, which I believe was an excellent introduction to me on open-source software and technologies. His suggestions greatly improved the quality of the project.

I would also like to thank my mentors Mr. Prashant Sinalkar and Ms. Sashi Rekha for providing valuable insight and expertise, as well as assisting me in overcoming the several difficulties I faced during the course of this project. Their advice on the spam-detector significantly improved the filtering process, and the various problems they noticed in the website considerably helped me in the bug-fixing phase. I would never be able to finish the project without their support.

I would also like to show my gratitude to my fellow interns and peers, who were constantly there to clarify my doubts and recommend improvements to the project. Their help and support was of immense value to me.

Contents

Chapter 1

Introduction

The **FOSSEE Forums** is a website which provides a platform for professionals and beginners alike to discuss a variety of technology-related areas by posting questions and answers. FOSSEE stands for 'Free and Open Source Science and Engineering Education'. The forum focuses on open-source technologies accessible to the general public. Users can choose to create an account to post anything on the forum, but doing so is not necessary, and they still have access to the entire forum content without logging in.

The strongest facet of the forum is its focus on niche areas and software. Experts in these areas are small in number, since many of these areas require multidisciplinary knowledge extending beyond only technology. For instance, DWSIM is a chemical process simulator which would require knowledge of chemistry, and OpenFOAM is a toolbox for mechanics and Computational Fluid Dynamics (CFD). It's relatively difficult to find people on the internet who can provide quality solutions to such problems, and that is where FOSSEE experts come in. They have knowledge in such niche areas, and can access the forum to answer questions posted on the website.

This summer fellowship project focused on primarily three areas: **bug fixes**, a **moderator interface** and a **spam filter**. The spam filter predicts if a posted question or answer is a spam post, and only the posts marked non-spam are visible to regular users. The moderator interface is a specialised section of the website accessible only by FOSSEE moderators, who are offered an array of features, giving them complete control over the forum posts. Since moderators cannot have knowledge in every area, they're usually assigned specific FOSS (Free and Open Source Software) categories, and posts outside these categories remain inaccessible to these moderators. This report will include three distinct sections on the project — the **User Section**, the **Moderator Interface** and the **Spam Filter** — outlining the working of and my contributions to each section.

The project required working on **Python** and **Django** (a Python-based web framework) for back-end management, **HTML, CSS, Javascript and JQuery** for front-end management, and **SQL** for database handling of the website.

Chapter 2

User Section

The User Section is accessible to anyone who visits the forum, regardless of whether he/she is a moderator or not. **Only the non-spam posts are displayed**. Without logging in, one can view all the questions, answers and comments, but cannot post anything on the forum.

2.1 Home View

The home view is the first web-page a user comes across on entering the forum website. It is relatively simple, with a carousel to show the various categories and an option to view all the questions in the forum. It also has a list of the most recent questions to make navigation a little easier.

My contribution here was to change the displayed questions so only the questions marked non-spam as visible to a regular user visiting the forum. As we will see, this changes when a moderator is logged in, who is able to view all the questions.

2.2 Filtering questions by category

The website has an option to filter all the questions by their respective categories. This can be done from the carousel in the homepage or by clicking the category of a particular question.

A previous bug was fixed in the all questions web-page where questions from the 'Sandhi' category were not displayed. Another bug was fixed which did not show the category name in the page title. Only non-spam questions are displayed.

2.3 Search bar

The web-page earlier consisted of a Google search bar, which did not align with how a forum search option should work. Moreover, the search option was present only in the home page, which made accessing it a little more tedious.

Removing the Google search bar, I implemented a search option in the main navigation bar, allowing users to search questions whenever needed. Questions are filtered according to their respective titles.

2.4 Asking a new question

Earlier versions of the project included an 'Ask A Question' option, but this was riddled with several bugs. These included allowing empty titles, problems in category selection, allowing descriptions with only HTML tags, and no option to upload an image.

Most of the errors existed in `NewQuestionForm` class in `forms.py`. Using the `BeautifulSoup` library, I added form validation checks, such that titles and descriptions which are empty or contain only HTML tags are not allowed. The `NicEdit` plugin was replaced by `CKEditor` for better formatting, and also because the former was blocking the submission of POST requests. Finally, an `ImageField` was added in `Question` model and `NewQuestionForm` for uploading an image to support the question. Image dimensions are restricted using the `django_resized` library and file size is also restricted to a custom value in `settings.py`.

2.5 Notifications

Whenever an answer is posted, the respective question creator is notified. Same happens when comments are posted to a particular answer. The users are notified using the `Notification` model, which consists of a user ID, question ID, answer ID and a comment ID to correspond to any particular user and post. Whenever the user logs into the forum, notifications with his/her user ID are available.

My contribution here consisted of fixing a major security concern, through which a user could manually enter a URL to access another user's notifications.

2.6 Viewing profile and changing password

Extending on the Django `User` model, the website uses a `Profile` model, adding additional attributes like phone number, confirmation code (for logging in the first time) and address. During local testing, any `superuser` created through the terminal does not have a profile, and 'View Profile' section gave errors in such cases. Fixing it involved creating an empty profile for a superuser, which needed to be provided to access certain features of the forum.

A significant option the forum initially lacked was the feature of changing passwords. If an unauthorized person acquires the user's password in such a case, he/she can access the account for extended periods of time. Adding a 'Change Password' option using Django's built-in views gives users the freedom to change their passwords whenever required.

2.7 Viewing a particular question

Perhaps the most obvious section of any forum, this section shows details of a particular question, along with the corresponding answers and comments. After all the answers is a form to upload your own answer, available only when a user is logged in. Details like the post creator, the date of creation and the question category are all mentioned.

The first bug-fix involved the upvote and downvote buttons, in which a user who's upvoted the post cannot downvote it, and vice versa. Moreover, in some cases, a user could simultaneously upvote and downvote a post. Fixing this involved modifying the `vote_post` and `ans_vote_post` views, as well as the JavaScript sections of `get-question.html`. Other improvements included giving an edit/delete question option to its creator, but only when no answers have been posted.

Clicking on the 'Edit' option now leads to a separate form similar to `NewQuestionForm`, but with the question's data already loaded. After submitting the edited question, the spam filter runs to predict if the edited question is spam/non-spam. Editing the question sends an email to the forum administrators, and the question creator, notifying them of the same. Selecting the 'Delete' option launches a confirmation modal, preventing unintentional deletion. Further improvements included replacing `NicEdit` in `AnswerQuestionForm` with `CKEditor`, and adding an image upload option to answers, by modifying `AnswerQuestionForm` and `Answer` model in a way similar to the one in section 2.4.

2.8 Logging in and editing profile

While an anonymous user can explore the forum without logging in, posting any content is reserved only for users having a FOSSEE Forums account. A minor bug-fix included fixing the redirection where logging in took the user directly to the home-page when the target page was a different one. If an anonymous user chooses to post anything, he/she is first redirected to the login page.

An important concern in the website was that users could circumvent editing their profile, resulting in accounts without any names, addresses or phone numbers. The solution included checking if a user has a fullname through a custom function, and using Django's built-in `@user_passes_test` decorator to allow selective access to certain sections of the forum. If any user chooses to post content without providing their full name, the website redirects to the profile editing page.

Chapter 3

Moderator Interface

The moderator section is available only to privileged users, who have been assigned as FOSSEE moderators by the website administrator. The administrator does so by adding certain users to moderator groups in the Django admin panel. Moderators are assigned specific FOSS categories corresponding to their expertise, and to prevent compromising the quality of content, they do not have access to the remaining categories. A moderator has complete control over the forum content, with features like deletion/editing of questions and answers, and deletion of comments. **All posts regardless of their spam property are visible to moderators**, and a moderator can mark posts as spam/non-spam. A regular user has an extra 'Moderator Panel' option in the navigation bar, provided they're assigned as moderators.

This section relies on two concepts. The first is an `@is_moderator` decorator, which provides selective access to web-pages by checking if user belongs to any moderator group(s). The second is a `MODERATOR_ACTIVATED` variable in `settings.py`, which controls the base template and displays different navigation bars for a regular user and a moderator.

3.1 Home View

The moderator home view works in a similar way as the normal home page does. The only differences are that instead of a carousel with all categories, only the categories pertaining to the moderator are displayed. Moreover, the recent questions table has an extra field displaying the spam property of a question. The total question count in the home-page is modified to count only the questions pertaining to moderator's FOSS.

Moreover, the 'Ask a Question' option has been removed from the carousel and the navigation bar, because a moderator (unlike a regular user) is only supposed to monitor and answer the existing posts and not ask new ones. The `MODERATOR_ACTIVATED` variable becomes active on entering this page, so now the default navigation bar changes, and the original one is seen only on going back to the User Panel.

3.2 Viewing a particular question

Like its corresponding section in the User Panel, it shows the details of a particular question, along with a list of answers and comments. The difference is that a moderator now has complete privileges to delete and edit any post in the page.

On the top-right corner, the edit and delete options for the question are visible. 'Edit', just like its previous counterpart leads to a new form similar to `NewQuestionForm`. 'Delete', on the other hand, works a little differently. It launches a confirmation modal with a textbox, for specifying the reason of deletion. The moderator can then specify the reason for question deletion, so that the user can re-post a question with better quality. A spam/non-spam badge is added to the description, and clicking on it leads to a web-page with all the spam/non-spam questions concerning the moderator.

Each answer has a delete and edit option in the top-right corner. The 'Edit' option loads an in-page form with `NicEditor`, and the editing of answers is handled dynamically using the AJAX (Asynchronous JavaScript and XML) view `ajax_answer_update`. The 'Delete' option works in the same way as the above mentioned 'Delete' option for questions. In addition, a spam/non-spam badge is added along with an edit option, which generates a modal to modify the spam property of an answer. The modal uses the HTML element `<select>` to provide a drop-down list of options. The comment section uses a hover button, where if the moderator hovers over a particular comment, a delete option appears. This is to reduce clutter in the web-page. Since this is the moderator section, no forms for comment or answer upload are available.

3.3 Editing a question

The 'Edit Question' option of the moderator section differs in subtle ways from that of the User Section. The first difference is that it has an extra check-box option to mark our question as spam/non-spam. Since the same form (`NewQuestionForm`) is used to edit a question in both user and moderator section, the spam check-box appears only when `settings.MODERATOR_ACTIVATED` is true. The other difference is that since this is a moderator editing a question, he/she is given preference over the spam filter, and the question's spam property is decided only by the check-box.

3.4 Spam, non-spam and unanswered questions

The navigation bar provides options for viewing spam, non-spam and unanswered questions. These extend from the 'View all questions' section, with the spam/non-spam sections having an extra parameter in their URL. The unanswered section filters its questions in the HTML code using the condition `question.answer_set.count() == 0`.

Chapter 4

Spam Filter

A major component of the project was developing a spam filter, which would go through any content before it is posted, predict if it is spam or not, and assign the spam property to every post. With the internet manifesting in every aspect of our lives, it has become extremely easy to broadcast content to a wide array of people. This is a double-edged sword, because while it makes access to information easier, it also leads to the proliferation of unwanted and irrelevant data. This can be especially pronounced in a forum, where all users need to do is make an account to post content.

Spam data can clutter websites, reducing focus on what is important and relevant. It leads to wastage of user's time, affects productivity, and in the case of a forum, reduces a user's trust. With this in mind, it is very important to reduce the spam data posted on the website.

4.1 Reading the available data

It is true that the best machine-learning (ML) program is not the one which has the best algorithm, but the one which has the most data. Data collection is perhaps the most important part in developing an ML-algorithm, and significant portions of time are spent on the same.

The best data we could get was the data from the forum itself. Using a training dataset (a `.xlsx` file) given for the screening task, as well as all the forum questions and answers, an `xData` list is constructed for the data itself, and a `yData` list for the spam property. Reading the forum ensures that all kind of data is considered, and using an additional `.xlsx` file allows the administrator to add custom data for the ML-algorithm to train upon.

For reading data from the `.xlsx` file, the Python library `openpyxl` has been used. The spam filter is re-trained whenever a moderator marks a non-spam question as spam, and vice versa. It is also trained once each day using a cronjob in `cron.py`, with the function `train_spam_filter()`.

4.2 Data pre-processing

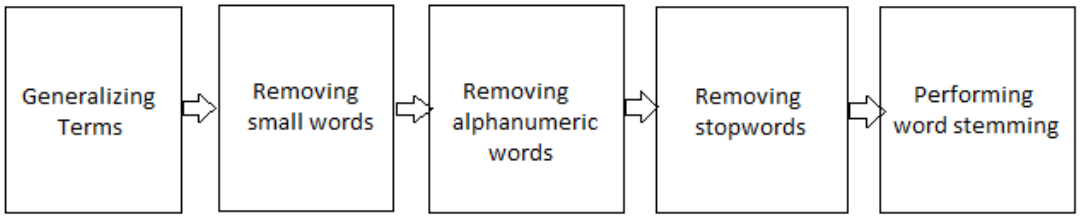

Figure 4.1: Steps of text pre-processing

In filtering of spam, the pre-processing of text is important for obtaining critical and uniform data. In this step, data which is redundant or not useful is removed, and words with the same root are generalized. Our data pre-processing involved generalizing terms, removing words with length less than or equal to 2, removing alphanumeric words. Finally, stopwords are removed and word stemming is performed to reduce the vocabulary. Python's regular expressions and the Natural Language Toolkit (`nltk`) library is used.

Word in actual data	**Processed word**
URL addresses	httpaddr
E-mail addresses	emailaddr
HTML `<a>` tags	linktag
HTML `<img>` tags	imgtag
Digits and numbers	number
$	dollar
!	exclammark
?	questmark
`\n`	newline
`\n\n`	blankline

Table 4.1: List of generalized words/terms

Other words are either removed if they are alphanumeric or in `nltk`'s stopword corpus. Remaining words are also stemmed using `SnowballStemmer` in `nltk` to reduce vocabulary. HTML tags apart from `<a>` and `<img>` are simply removed.

4.3 Machine-learning algorithms used to train on data

Training on the available data is the most important component of any machine-learning program. In our spam filter, training categorizes the kind of data which constitutes spam, and uses that categorization to predict future posts. A variety of algorithms are available for spam filtering, but the ones most prevalent are **Support Vector Machines (SVMs)** and **Naive Bayes Filtering**. Since the available dataset was way skewed, the F-score instead of the accuracy is used as a performance measure. I've used the `LinearSVC` library for testing SVM since it gave the best results in my case and others[1], and for Naive Bayes I've used the `MultinomialNB` library. A total of 233 examples are used with a train-test ratio of 3:1.

Model	Precision	Recall	F-score
`LinearSVC`	0.95	0.99	0.97
`MultinomialNB`	0.95	0.99	0.97

Table 4.2: Results with `CountVectorizer`

Model	Precision	Recall	F-score
`LinearSVC`	0.97	0.99	0.98
`MultinomialNB`	0.91	1.0	0.95

Table 4.3: Results with `TfIdfVectorizer`

As observed, the most uniform results are given when `LinearSVC` is used with a `TfIdfVectorizer`. A count vectorizer creates a simple dictionary of available words in the training corpus, with each word having an equal weight. In contrast, a TF-IDF (term frequency-inverse document frequency) vectorizer considers the number of times a word appears in spam/non-spam emails and adjusts each word's weight accordingly, making it a better algorithm in our use context.

4.4 Predicting posts as spam/non-spam

Any string is first pre-processed using `clean_string` function in `cleanText.py`, following which our tf-idf vectorizer constructs a feature vector for the string. The linear-SVM model then predicts if the string is "Spam" or "Not Spam".

[1] Agarwal, Deepak, and Rahul Kumar. "Spam Filtering using SVM with different Kernel Functions". International Journal of Computer Applications 136.5 (2016): 20-22. Web. 3 July 2018.

Chapter 5

Other modifications and fixes

A variety of other improvements and fixes were implemented on the original codebase. Since this project was forked from the Spoken Tutorial repository, it contained a lot of code pertinent only to spoken tutorial. Several HTML templates and backend files contained instances of `minute_range`, `second_range` and `duration` variables which was completely removed. Moreover, a significant amount of code was not being used at all in the website and this was removed altogether.

Earlier, the number of views on a question updated whenever its corresponding page was opened, regardless of the possibility that the same user can view the question multiple times. Solving this included adding a `userViews` field to the `Question` model, so a user can increase the question's view count by only one.

The entire project was migrated first from Python 2.7 to Python 3.6.5, and then from Django 1.9 to Django 2.0. The entire codebase was updated to conform to PEP8 standards. A `HoneyPotField()` from the `django-antispam` library was added to the forms to detect and block automatic spam spiders. Custom templates for a 404 error and an 'unauthorized' error were implemented, and a `robots.txt` file was created. A total of 326 test cases were written, for all the models, forms and views.

References

1. Chaitanya Baranwal. FOSSEE-Forum repository. GitHub. https://github.com/chaitanyabaranwal/FOSSEE-Forum.

2. Sharma, Anjali, Manisha, and Dr. Rekha Jain. "Data Pre-Processing in Spam Detection". IJTSE - International Journal of Science Technology and Engineering 1.11 (2015): 33-34. Web. 3 July 2018. http://www.ijste.org/articles/IJSTEV1I11008.pdf

3. "TfIdf." Wikipedia, Wikimedia Foundation, 3 July 2018, https://en.wikipedia.org/wiki/Tfidf

4. "Sklearn.svm.LinearSVC." 1.4. Support Vector Machines - Scikit-Learn 0.19.1 Documentation, https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html.

5. "Support Vector Machines." 1.4. Support Vector Machines - Scikit-Learn 0.19.1 Documentation, https://scikit-learn.org/stable/modules/svm.html.

6. "Sklearn.feature_extraction.Text.TfidfVectorizer." 1.4. Support Vector Machines - Scikit-Learn 0.19.1 Documentation, https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html.

7. "Feature Extraction." 1.4. Support Vector Machines - Scikit-Learn 0.19.1 Documentation, https://scikit-learn.org/stable/modules/feature_extraction.html.

8. "Porting Python 2 Code to Python 3." 4. More Control Flow Tools - Python 3.6.5 Documentation, https://docs.python.org/3/howto/pyporting.html.

9. "Upgrading Django to a newer version", Django Documentation, https://docs.djangoproject.com/en/2.0/howto/upgrade-version/.