Stack Exchange: Can we make it better?

StackPhobia **15.04.2020**

Weijia Li (301395257) Nan Xu (301397230) Haoran Chen (301400745)

1 Motivation and Background

Stack Exchange is one of the most influential open question-and-answering (Q&A) communities in the world, which provides more than 170 different platforms each covering a specific topic. Its most visited website Stack Overflow, which is designed for programmers, has more than 11 million users and 10 million visits per day. Therefore, any imperfection of Stack Exchange might negatively affect many people.

Stack Exchange requires users to label their questions with tags, which are short descriptions less than a few words long. In order to improve the accuracy of tags, Stack Exchange allows other users to edit them after the posting. As the results of an initial analysis, we notice that users often have trouble selecting the most accurate tags for their questions. There is one post even having its tags edited by others for at least 74 times, since it was created. The problem of initial tags not being trustworthy makes it more difficult for others to find the relevant information. And for users who are willing to answer questions, if posts recommended to them are based on the wrong tags, they might not be able to answer them. In the end, it takes longer to get the answer for a question.

Besides, while checking the contents of some posts, we happen to find out that there is offensive or impolite language used, not only in the posts, but also in the comments. We believe Stack Exchange is a professional discussion platform. So it will be nice to have such a functionality to prevent users from posting offensive languages, making Stack Exchange a better discussion place and improving user experience as well.

Last, we have been quite interested in finding the relations among different platforms. Can we correlate users' inter-platform interests among all Stack Exchange communities? Do the users who are interested in one topic on one platform show interest on another platform? Is it true that engineers interested in computer vision are also fans of climbing? We are here to perform cross-platforms analysis and try to uncover any hidden relation. Such correlation can give Stack Exchange a boost in business. Compared with Stack Overflow, other platforms such as Music, Gaming and Bicycle, have far fewer registered users and posts. What if we use such relations to promote other platforms?

1.1 Related Work

Over the recent years, many attempts have been made to make Stack Exchange a better community since it was created in 2008. In 2013, Stack Exchange even offered an experimental tag suggestion system for a few of its sub-platforms, but it has never been rolled out to all sites due to performance issues. Earlier methods for predicting the tags in Q&A communities, including the one used by Stack Exchange are mainly based on Bayesian inference and all suffer from poor performance [1]. To our knowledge, no attempt has been made to detect offensive language within Stack Exchange. Most closest work includes detecting hate speech on Twitter [2] and classifying toxic comments for Wikipedia [3]. There is also no previous work on analyzing users' inter-platform in Stack Exchange.

2 Problem Statement

In order to make Stack Exchange better, we focus on three questions:

1. Tag Recommendation

   Can we improve the accuracy of tags? Instead of manually selecting tags, can we automatically recommend the most accurate tags to users, whenever they post a new question?

2. Offensive Language Detection

   Can we develop a politer discussion community, via identifying and filtering offensive language, and imposing restrictions during the posting phase?

3. Cross-Platforms Analysis of Interests

   Can we uncover the hidden relation between users' interests among all Stack Exchange communities? Do the users who are interested in one topic on one platform show interest on another platform?

2.1 Challenges

Our project is full of challenges. First, we are dealing with large datasets. Stack Overflow alone has 260 GB data. It is both challenging to store all the data in spark dataframes or Cassandra tables. Since we are performing cross-platform data analysis, data integration can be super complex. At the same time, this project requires strong knowledge of Data Science, Natural Language Processing and Machine learning, especially Convolutional Neural Network and Recurrent Neural Network.

There is no annotated Stack Exchange data for training an offensive language detection model, and it is not practical to label Stack Exchange manually. The training process is also bounded

by hardware. Given the extreme large datasets, training and fine-tune of the tag prediction as well as the offensive language detection model require massive computing powers. Doing it without a GPU is time consuming if not impossible. Once our models are ready, deploying them as a usable web service can be challenging, since such services often require in-time prediction.

3 Data Science Pipeline

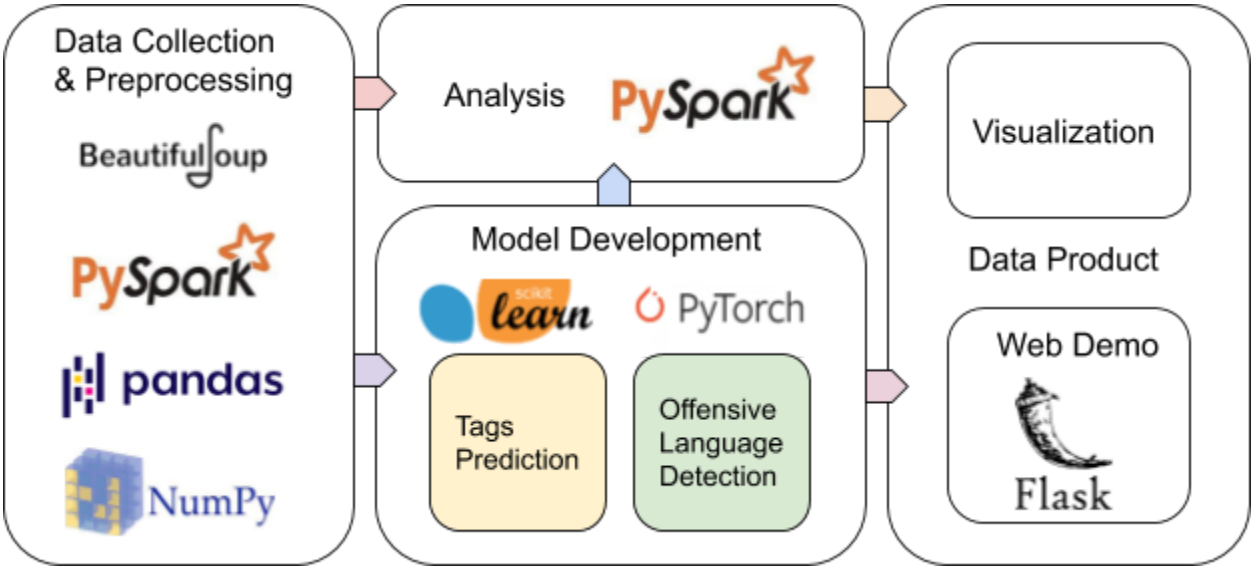


Figure 1. Our data science pipeline has three parallel steps in the middle.

Our project is mainly based on the data from Stack Exchange archive, which has historical data from Stack Overflow and 170+ other community-powered Q&A sites. In this project, we choose five platforms in their unique areas, including programming, bicycles, gaming, movies, and music. Beside using PySpark to extract users and posts data from these platforms, we also scripted tag synonyms from the stackexchange tag websites using beautifulsoup.

The preprocessed data then go into two different tasks. On one hand, we further integrate data from five sites together and perform cross-platform analysis. On the other hand, we use preprocessed data as ground truth to train our tag prediction model.

At the same time, we trained an offensive language detection model with dataset from twitter and wikipedia. During the analysis phase, we use this model to detect any offensive or impolite languages appearing in stack exchange communities.

Our final data product includes a tag prediction model, a offensive language detection model, and visualizations for any insight we found during the cross-platform analysis and our evaluation

for offensive languages in the communities. We also deployed both t models as a web service, to demonstrate the possibility of controlling post quality with automation.

4 Methodology

4.1 Tag Prediction

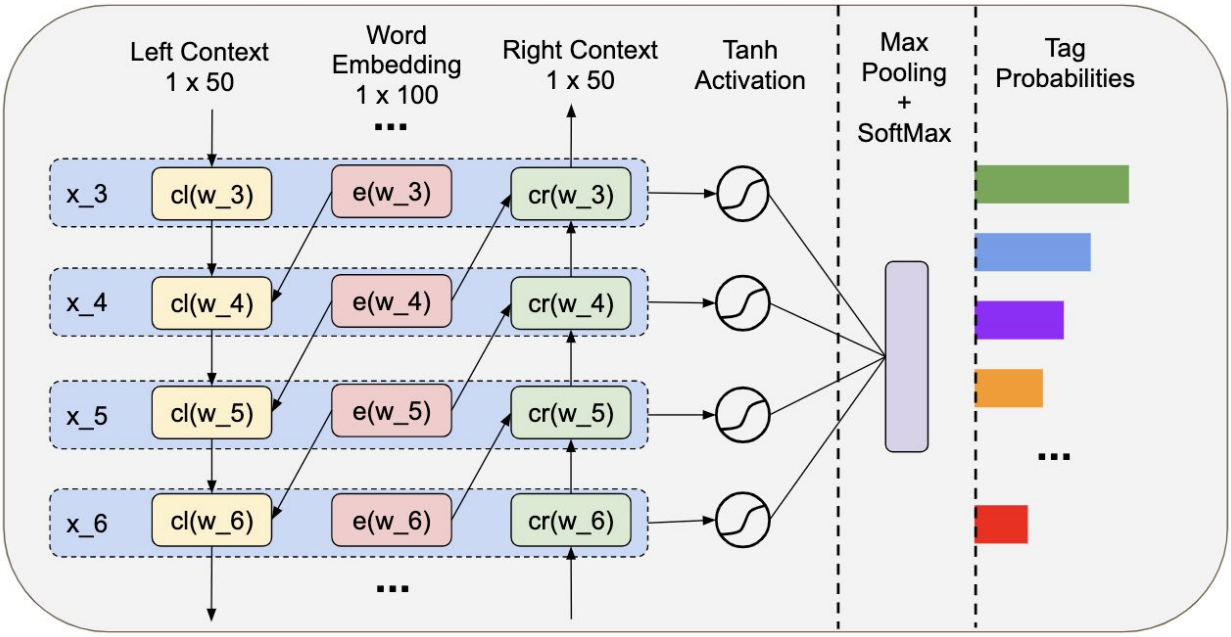


Figure 2. The tag prediction model we used assembles a bi-directional structure.

The tag prediction model we used is a neural network with bi-directional recurrent and convolutional structure [4]. We use the combination of title and message body as the input features. Given an input word sequence (w_1,w_2,w_3, …w_n), we represent each word w_i with three vectors: its left context cl(w_i), its embedding e(w_i), and its right context cr(w_i). This bi-directional structure allows the network to capture the contexts of a wordse through learning.

After concatenating these three vectors to form the final representation x_i of a word, it applies a linear transformation together with the tanh activation function to x_i. In this step, the most useful factors for representing the sentences will be selected. A max pooling layer right after it further narrows down the most important area of interest.

The final softmax layer outputs the probability distribution over all possible tags. We then select the 5 tags with the highest probability ranking as our predictions.

4.2 Offensive Language Prediction

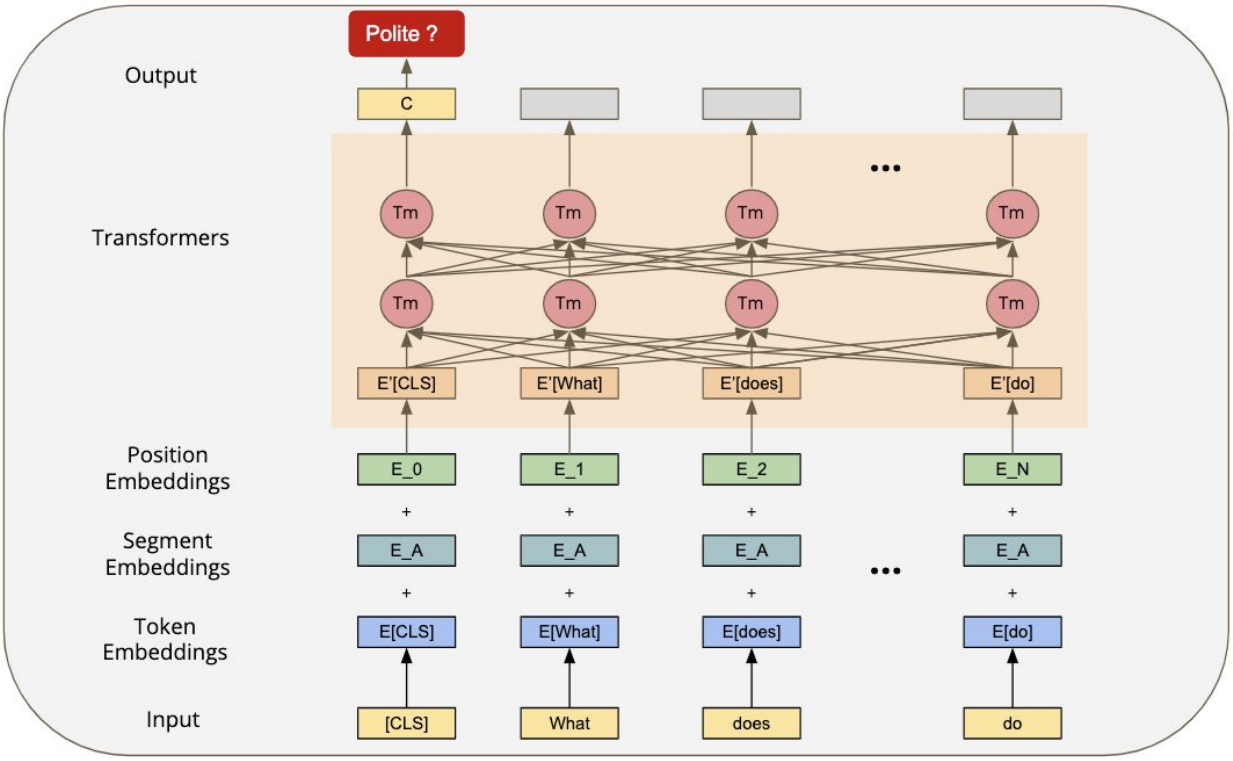


Figure 3. We modify the structure of BERT to fit our problem.

The offensive language detection model we used is based on BERT [5], the famous pretrained bi-directional transformers. BERT stands for Bidirectional Encoder Representations from Transformers, can be fine-tuned per task and achieves state of the art performance in multiple NLP tasks. We chose to fine tuned it for the offensive language detection task.

Like any other BERT version, each word in the input word sequence is embedded as 3 separate vectors before feeding into the transformers. These 3 vectors are position embedding, segment embedding, and token embedding.

But this model is still different than others in many ways. The most noticeable change is made in the output layer. Since we only want to know whether the sentence contains offensive language, the model output is a single probability number, instead of the original multi-element vectors.

The input is also different. It only consists of one segment, instead of two segments like context and question pairs in question answering tasks. Correspondingly, the segment embedding is unified.

4.3 Cross-Platforms Analysis

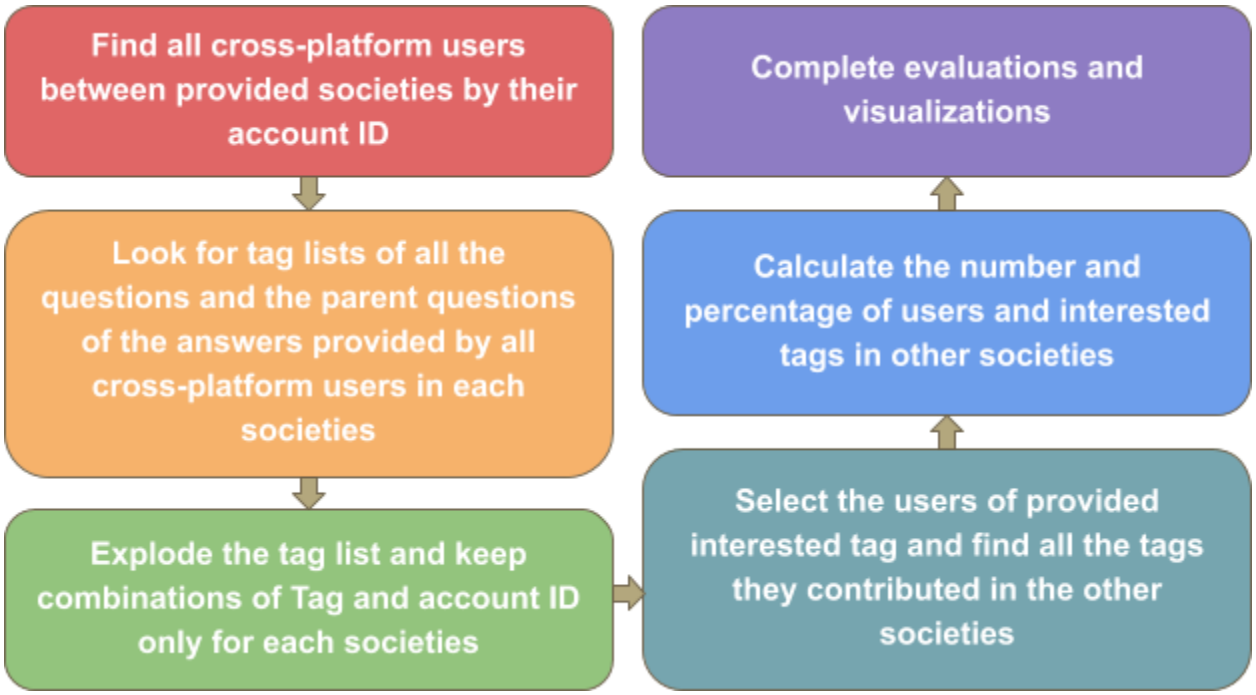


Figure 4. The inter-platform analysis is a 6-step process.

Considering this part being pure big data processing, we decided to use Pyspark running on the cluster to evaluate the data parallelly. The raw dataset is about 90 GB. It is a huge challenge for both storing and processing all the data in pyspark dataframes.

We first targeted the questions we wanted to solve and decided to only store several columns from the raw datasets. This process significantly reduced the space we needed to save the data, as some of the large columns like the body and LastEditDate in the posts are dismissed. We then had a discussion on whether we should store the data from one data file into several small blocks to boost up the speed of saving and loading data in further processing. We also discussed if we need to denormalize the data during preprocessing to avoid any joining later on.

After doing some research on Pyspark and some local tests, we found the smaller blocks did not influence much to the loading speed so we kept the original files, and because of PushedFilter and Broadcast Join, we did not need to worry much about the usage. After optimizing the order of processing, every file was only loaded once, and as we saved several checkpoints, the program was robust enough for any cluster or server crush. Moreover, there was no hardcoding anywhere in the program, so that we can easily expand the code to more complex projects or APIs.

4.4 Tool Selection

- Spark - we use spark for both preprocessing and real-time user interactive query. It scales well with our 130 GB data and has fault tolerance built in.
- Flask - we chose to build web server with Flask web framework for two reasons: since we are already using pyspark and native python to handle the ETL and data gathering jobs, it makes sense to keep this consistence and use a python based web framework; flask is also one of the most popular web frameworks, which is well supported and is easy to use.
- AWS EC2: Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers. We utilized AEW EC2 Deep Learning AMI (Ubuntu 18.04) Version 27.0 (ami-0142046278ba71f25) for model training.

5 Evaluation

5.1 Tag Prediction

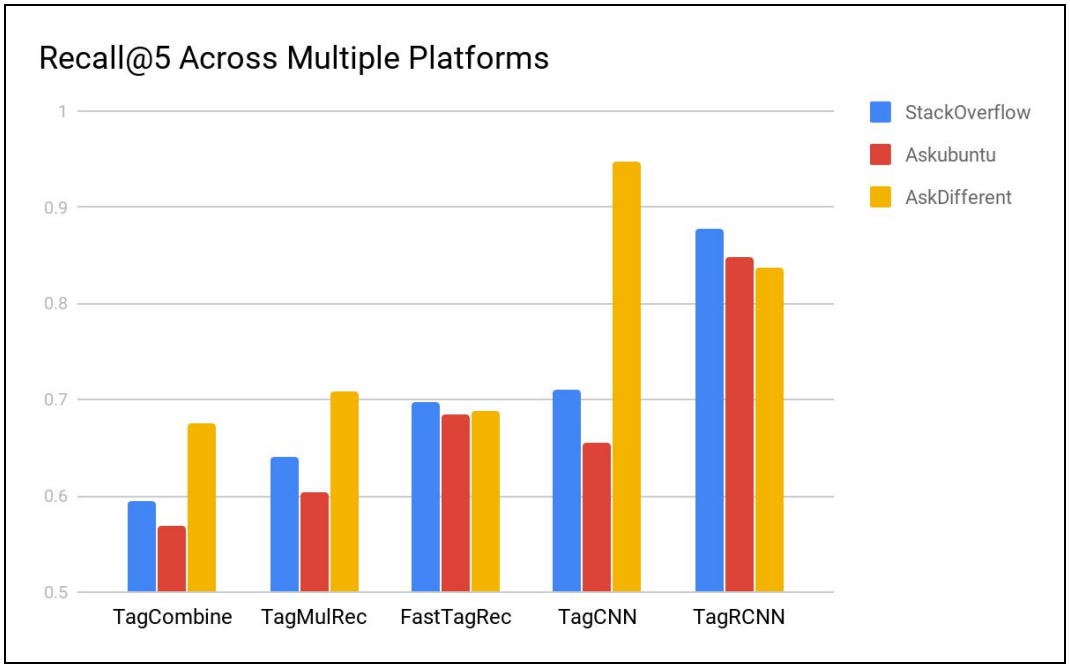


Figure 5. RCNN outperforms every other method.

Since Stack Exchange allows a maximum of 5 tags for each post, we use recall@5 as the evaluation metric. Here the recall@5 represents the percentage of true tags that matches the model's predictions assuming we only let the model to output 5 tags.

In the model selection phase, we experimented with 5 state-of-the-art tag prediction models with different approaches as shown in Figure 5. Because Recurrent-Convolutional Neural Network outperforms every other method in our experiment, it is the best solution for this project.

Platform	Recall@5	Recall@10
StackOverflow	0.89	0.92
Bicycles	0.76	0.84
Arqade	0.91	0.94
Movies & TV	0.96	0.97
Music	0.82	0.89

Table 1. Our finalized models generate accurate tag predictions on every platform.

As shown in Table 1, our finalized models can generate accurate tag predictions on every platform. With 0.8 recall, the model can successfully recommend 4 out of 5 tags, which significantly reduce the need of manual tag correction. The relative low model performance in the bicycle community might be caused by the small dataset. We anticipate a more accurate tag prediction model as the community grows.

5.2 Offensive Language Detection

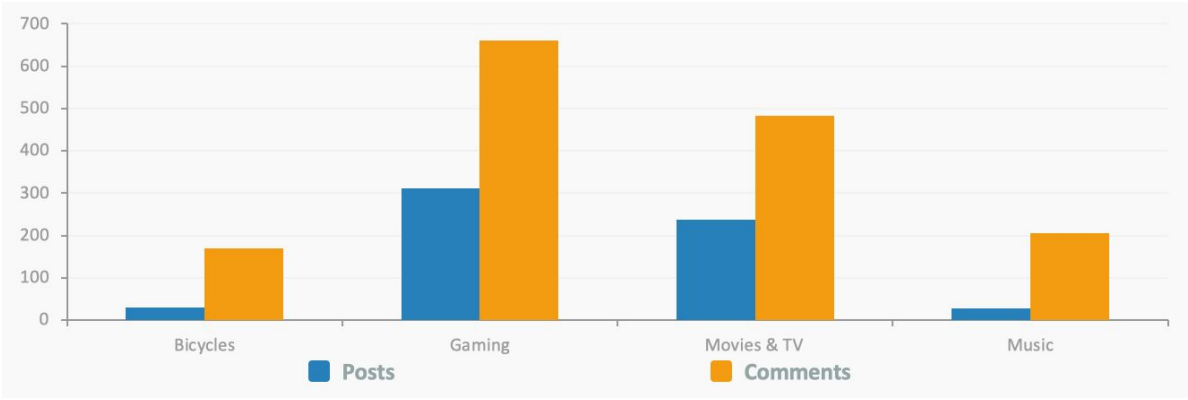


Figure 6. Number of occurrences of offensive language.

The offensive language detection model achieved a test accuracy of 93%. After deploying the model on the Flask web application, the prediction only takes 0.001s and it can detect offensive language in the text successfully. We also use the trained model to analyze all the posts and comments in the four platforms listed in Figure 6. The number of posts and comments having offensive language are shown in the chart.

5.3 Cross-Platforms Analysis

As this part is an experimental expansion to the project, we do not have a baseline result to compare with. To improve the accuracy of our results, we replaced the tag synonyms with the most common used one. It helped us decrease the number of tags we monitored and connected more users with the given tags.

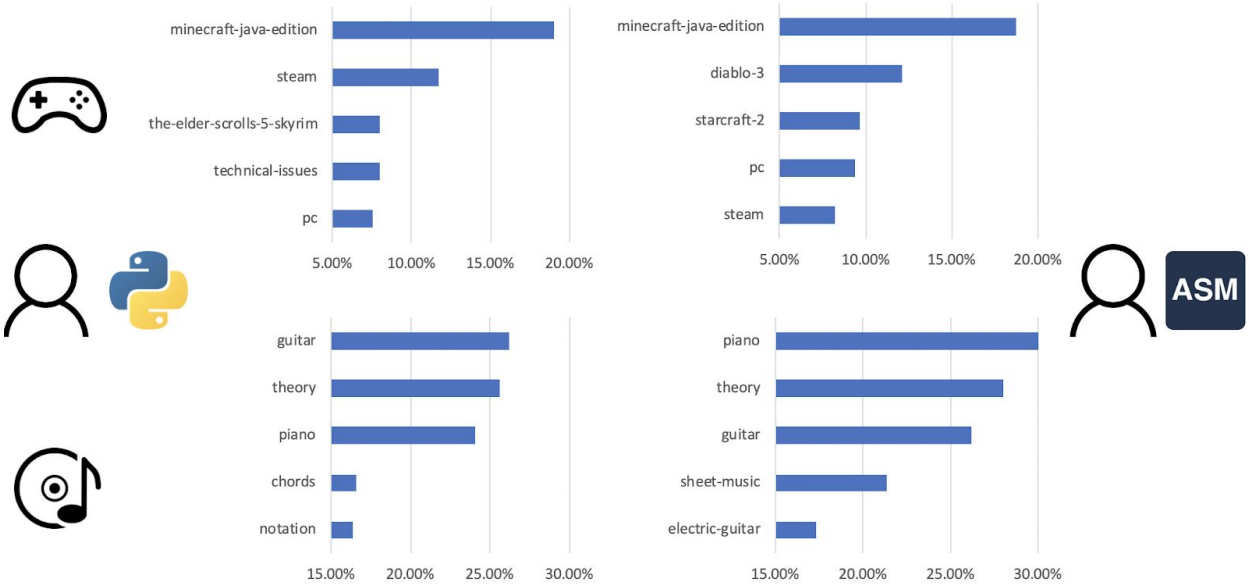


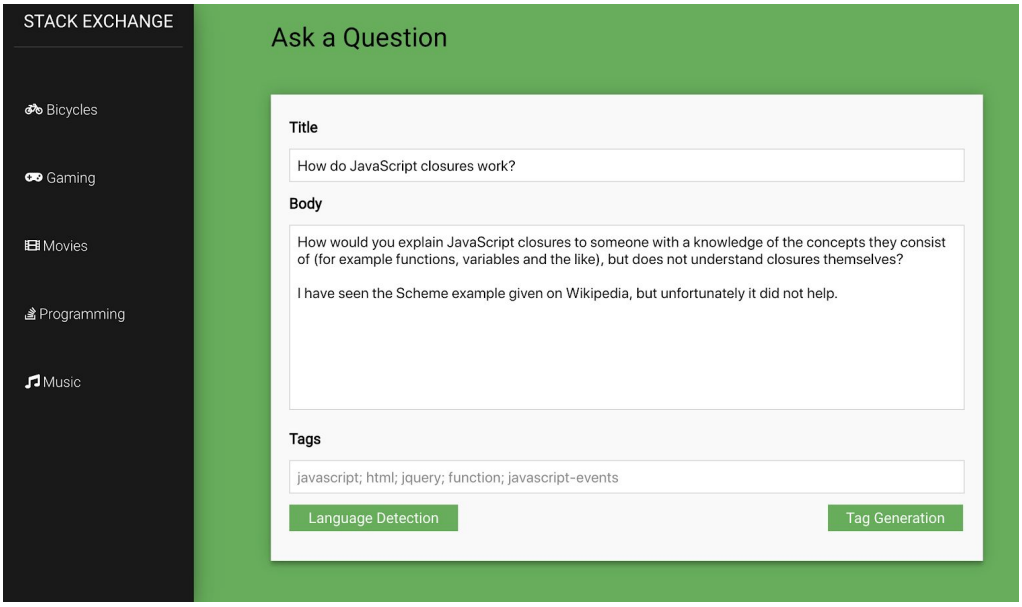
Figure 7. The output of popular tags in games and music societies among different groups of users.

We tested a number of tags from stackoverflow, and we found the output with tag “big-data” extremely close to the tag “python”, but “assembly” tag provided distinct results. In general, most big data topics are somehow relevant to python, but assembly programming is completely another topic. Therefore, the similar results from python and big data and the various popularities from assembly programming showed what we expected.

Moreover, based on the society culture, the questions and answers being kept and recommended came from the core users of the societies. So that, we believe these tags collected from the questions and answers should highly represent the common behaviour of the cross-platform users.

6 Data Product:

6.1 Web Services



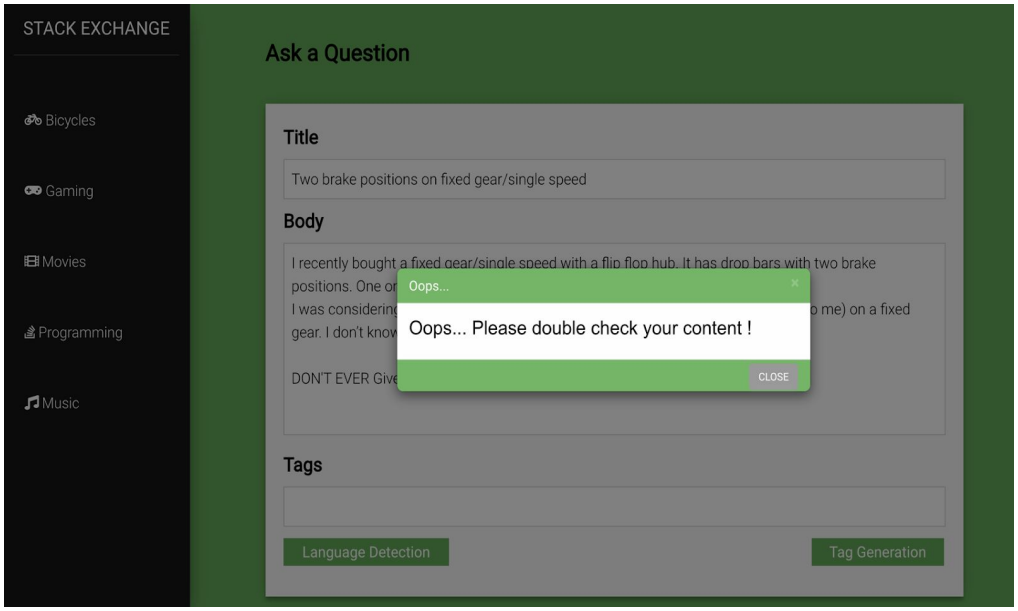
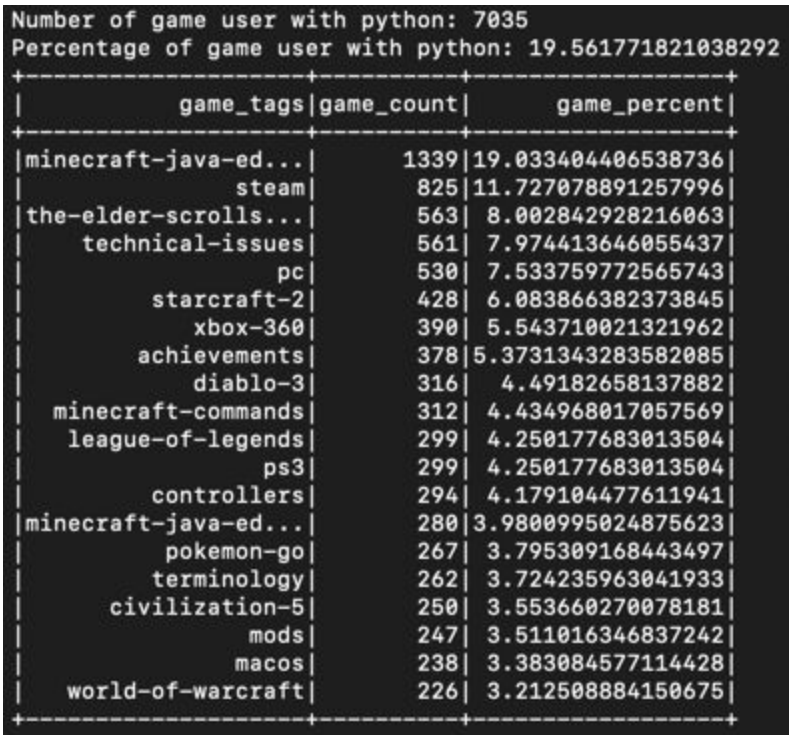


Figure 8 and 9. A Flask Application hosting Tag Prediction and Offensive Language Detection Model.

We created a web application using Flask and AJAX, and then deployed the tag prediction model as a web service. Whenever users want to post a new question, 5 tags that categorize their question well will be generated for them after a simple click of the button. This process takes less than 0.1 second.

We also deployed the offensive language detection model in our web application as well. After entering the message and clicking the button, it will run the detection model for inappropriate language and get the results in less than 0.1 second. If any offensive language is detected, it will popup an alert box. Otherwise, the message will be published successfully. Figure 8 and 9 demonstrate how it works.

6.2 Cross-Platforms Analysis



```
Number of game user with python: 7035
Percentage of game user with python: 19.561771821038292
+--------------------+----------+------------------+
|           game_tags|game_count|      game_percent|
+--------------------+----------+------------------+
|minecraft-java-ed...|      1339|19.033404406538736|
|               steam|       825|11.727078891257996|
|the-elder-scrolls...|       563| 8.002842928216063|
|    technical-issues|       561| 7.974413646055437|
|                  pc|       530| 7.533759772565743|
|         starcraft-2|       428| 6.083866382373845|
|            xbox-360|       390| 5.543710021321962|
|        achievements|       378|5.3731343283582085|
|            diablo-3|       316|  4.49182658137882|
|  minecraft-commands|       312| 4.434968017057569|
|   league-of-legends|       299| 4.250177683013504|
|                 ps3|       299| 4.250177683013504|
|         controllers|       294| 4.179104477611941|
|minecraft-java-ed...|       280|3.9800995024875623|
|          pokemon-go|       267| 3.795309168443497|
|         terminology|       262| 3.724235963041933|
|      civilization-5|       250| 3.553660270078181|
|                mods|       247| 3.511016346837242|
|               macos|       238| 3.383084577114428|
|   world-of-warcraft|       226| 3.212508884150675|
+--------------------+----------+------------------+
```

Figure 10. An example evaluation of game society from the users interested in the “computer-vision” tag.

The fundamental outputs for this part are provided by calculated results shown in sentences and tables separated by the societies. The users are allowed to provide any societies to explore in Stack Exchange platform, and all the tags from the base platform are supported. The evaluation for the behaviours of users interested in a given tag to other societies after processing the datasets is around 10 seconds.

Here we provided an example of using stackoverflow as our base society, while using bicycles, gaming, movies and music as testing societies. The number of users in each society with a given stackoverflow tag, and the percentage of the users posted in the society by all the users with the given tag are provided. Then in detail, a full list of the tags in the society is given in the popularity order. The count and percentage of all tags mentioned in the posts in the societies collected from the cross-platform users are also provided in the tables.

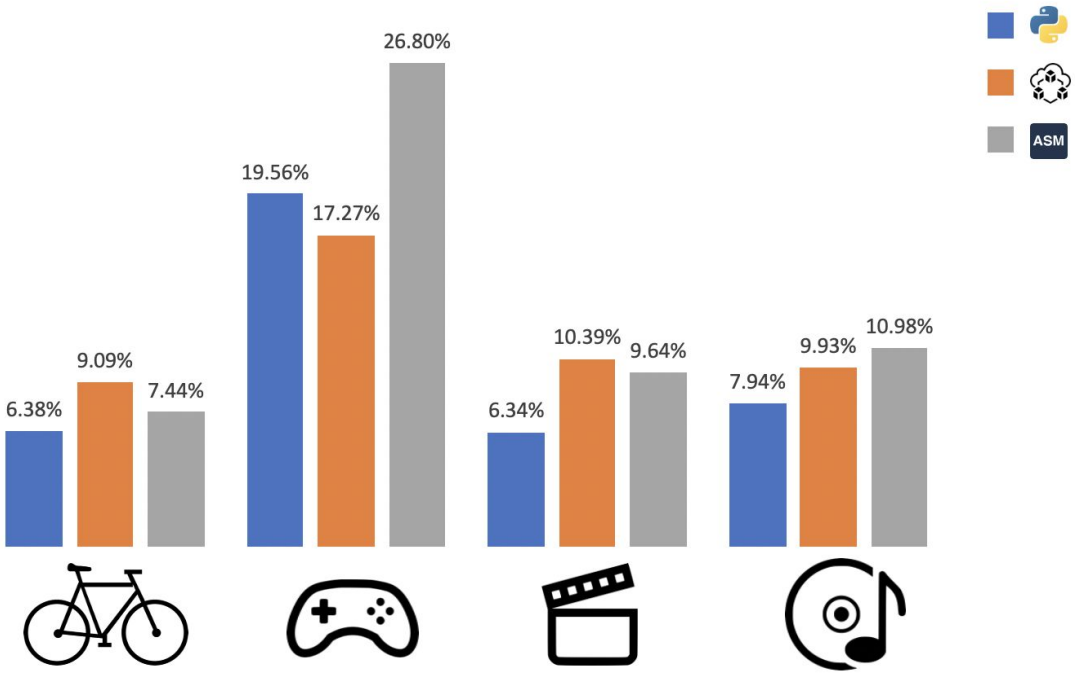


Figure 11. Percentage of user asked or answered questions in each society based on interests.

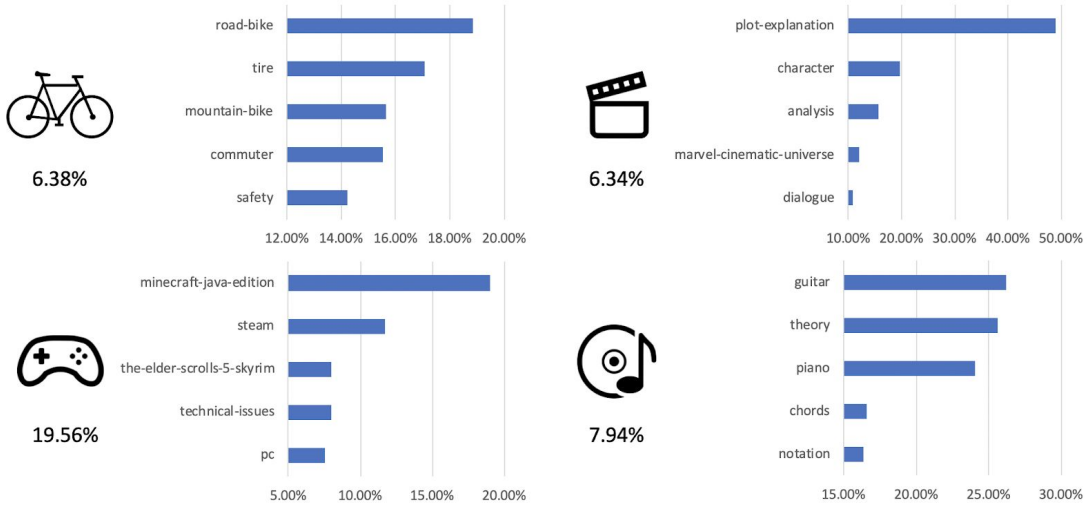


Figure 12. Five most popular tags in each platform with python users.

From the result, some visualizations can be explored as well. We compared the ratio of different groups of cross-platform users who contributed to the societies and provided the different preferences of python users and assembly programmers in games and music societies as some examples.

7 Lessons Learnt

- Dive into the state-of-the-art research on tag recommendation and offensive language detection.
- Learned text preprocessing techniques for Natural Language Processing related tasks.
- Mastered the structure of a bi-directioanl recurrent and convolutional network.
- Learned how to fine-tune a BERT model for offensive language detection tasks.
- Learned the importance of processing order and optimization in data processing.
- Learned that the correct usage of filtering and combination of pyspark.sql.functions extremely decrease the running time of pyspark.
- Writing codes in generic purpose is painful during programming, but it is significantly easy and expandable when being used with other codes and further possible APIs.
- Built a Flask powered web service offering in-time model prediction.

8 Summary

Like millions of people who use Stack Exchange everyday, we like Stack Exchange and want to make it better. Working toward this goal, we identify three potential areas of improvement, including inaccurate tag selection, offensive language within the community, and the lack of inter-platform analysis. We solved the first two issues with a tag prediction model based on RCNN, and an offensive language detection model based on BERT, respectively. Lastly, our analysis on the inter-platform user interests provides a unique way to boost user activities. We believe our work can benefit both Stack Exchange and millions of its users.

References

[1] Wang, Shaowei, et al. "EnTagRec++: An enhanced tag recommendation system for software information sites." *Empirical Software Engineering* 23.2 (2018): 800-832.

[2] Robinson, David, Ziqi Zhang, and Jonathan Tepper. "Hate speech detection on Twitter: feature engineering vs feature selection." *European Semantic Web Conference*. Springer, Cham, 2018.

[3] Georgakopoulos, Spiros V., et al. "Convolutional neural networks for toxic comment classification." *Proceedings of the 10th Hellenic Conference on Artificial Intelligence*. 2018.

[4] Zhou, Pingyi, et al. "Is deep learning better than traditional approaches in tag recommendation for software information sites?." Information and software technology 109 (2019): 1-13.

[5] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).