G5291 Advanced Data Analysis

Final Project: **Stack Overflow Q&A Analysis**

Group 2

Duanhong Gao (dg2896)
Yi Jian (yj2376)
Jingwei Li (jl4549)
Aoyuan Liao (al3468)
Hanqing Shi (hs2871)
Jia Wang (jw3315)
Xiangyu Wu (xw2423)
Yutong Yang (yy2624)
Wanyi Zhang (wz2323)

Columbia University
Fall 2016

Contents

Chapter 1

Introduction

Stack Overflow Q&A Analysis

To begin with, we try to analyze the differences between R and Python questions in terms of popularity, hot topics, response time, and the relationship between scores of the question and its answers.

The second part we try to predict question scores based on some features. From part 1's conclusion, we all hope that our problems can get higher score so that it can be resolved better. But how to make it happen? Here we are interested in two response variables: the score of the question and whether a question gets resolved.Then we applied exploratory data analysis to the score and time then we fitted logistic regression on score and time respectively. We also did xgboost to find out important features towards score and time.

The last part of our project aims to use machine learning methods to recommend tags for each question asked on Stack Overflow. We use Latent Dirichlet allocation (LDA) to do topic modeling to all R questions from Stack Overflow, which are described in full detail in the Date Description section. Then for each test question we apply k-nearest neighbors (k-NN) algorithm and ranking of the numbers of tags to recommend tags, finally, to make our result more accurate, we add weights to those tags which appear in the question bodies. The rest of the paper is organized as follows. Section III illustrates the rationale, which is the algorithms we choose and the reasons. Section IV presents the results of recommendation, including the plot of our topics and the most common words in each topic, a sample list of comparison of the actual tags and recommended tags, and the error rate of the whole test data set. Section V draws the conclusion, which states that our method is doing a pretty good job in recommending tags for questions. Also, we suggest possible improvement for the research in the end.

Chapter 2

Objectives

Our graduation is coming soon. As students in statistics, we know some of us are seeking for a career in data science. However, looking through long description of data scientist position, we found that data scientists seem to be highly rare 'awesome nerds'- those who embody the perfect skillsets of math and stats, coding, and communication. We, statstical students, don't need to worry about stats knowledge. However, the shortcoming for most of us, coding, is keeping us back at 'nerds'. Our motivation is to help us make most of Stack Overflow, improve coding skills and rush towards 'awesome nerds'. Here we want to answer three problems raised by the dataset, which are also our objectives.

1.What are the differences between R questions and Python Questions?

2.How can we increase the probability that we got acceptable answer for our question?

3.Would it be possible to recommend paired and regularized tags automatically?

Chapter 3

Materials & Methods

3.1 Data Source

We explored two datasets provided by Stack Overflow on **Kaggle Dataset**. We have two sets of data, one contains questions and answers from Stack Overflow that are tagged with ther tag, another one contains questions and answers from Stack Overflow that are tagged with thepython tag. Below is data description for first two parts of this project.

- R Questions from Stack Overflow
  - **Questions** contains the title, body, creation date, score, and owner ID for each R question.
  - **Answers** contains the body, creation date, score, and owner ID for each of the answers to these questions. The ParentId column links back to the Questions table.
  - **Tags** contains the tags on each questionbesidesthe R tag.
- Python Questions from Stack Overflow
  - **Questions** contains the title, body, creation date, score, and owner ID for each Python question.
  - **Answers** contains the body, creation date, score, and owner ID for each of the answers to these questions. The ParentId column links back to the Questions table.
  - **Tags**contains the tags on each questionbesidesthe Python tag.

In the last part of our project, we only use **R Questions from Stack Overflow** data set to do tag recommendation, since the methodology is the same for different data sets. And we only use **Questions** table and **Tags** table in this data set. For **Tags** table, we delete

tags which show up less than five times during the whole time period. We randomly split **Questions** table into training (80%) and testing (20%) sets.

3.2 Methodology

3.2.1 Exploratory Data Analysis

- **R & Python Questions Volumes per year**
- **Hot Topics Comparison**
- **Response Time Comparison**
- **Answer Score EDA**

Firstly, we want to draw a brief picture about how the volumes of the two languages questions varies over the last 8 years. We calculate the questions asked in per year in R and Python respectively. Then we visualize the hottest topics in two languages by word clouds. The word clouds show those hottest topics with their size corresponding to their frequency of being asked. Then. we look at the time it takes to have the first answer after a question posted. We apply exploratory data analysis to answer score and time variables to have a big picture of them.

3.2.2 Statistical Modelling

- **Robust Linear Regression**
- **Logistic Regression**
- **Gradient Boosting Trees: XGBoost**

Because of the departure of normality and outliers, we use robust linear regression to model the relationship between the score of each question and its corresponding highest answer's score.

From the histogram of the questions' scores, we can see that they are highly skewed. Based on that, we define that a question gets resolved when there is a corresponding answer with at least 3 scores, which is the 3rd quantile of the answer score distribution. Naturally, we applied a logistic regression to predict question's label. It shows that if the question score gets higher, the question will be more likely to get solved. Therefore, we can label each question as 0 or 1; 0 being a question without good answers and 1 being a question with good answers.

Due to the departure from normality, we appeal to non-parametric method, powerful XGBoost to find out what are the important features towards score and time. And

we divided them into groups according to their importance.Meanwhile, we extract some features from the raw dataset.Then, we train the model on these features. After tuning parameter, we got a cross-validated MSE for the best model of R Questions as nearly 8.2 and MSE for python questions as 19.1. From the importance matrix, we can see that, for R questions, sum_tag_freq, max_tag_freq, code_length are most relevant to the question score. And there is a gap of information gains between features in cluster 1 and features in cluster 2. As for Python questions, the most important factors change. Title length is inserted into the front row.

3.2.3 Tag Recommendation System

In order to recommend tags for questions, we need to find the most similar questions in training set, and choose the most common tags as recommendations. For each test question, we first do LDA topic modeling to cluster questions into 50 topics, and then use k-NN to find the nearest 20 questions to the test question, and rank the tags belonging to these questions. After that, we look through the body of the test question, add weights to tags which show up in the question body. Finally, we rank these weighted tags again and choose the most likely 6 tags as recommendations. More specifically, we use the following steps to do recommendation.

To begin with, we deal with our data as described in Section II to obtain the useful data for our algorithms. After this, we have a training data set and a test data set of questions. Secondly, we apply LDA to the whole question data set, and cluster these questions into 50 topics.

Thirdly, to find the nearest questions for each test question, we use k-NN algorithm to all training questions belonging to the same topic as that test question with a distance defined according to Jaccard index, which is the size of intersection divided by the size of the union of unique words in test question body and unique words in training set question bodies. By doing this, we obtain 20 nearest questions to each test question.

Finally, we rank the tags belonging to the 20 chosen questions. To improve our prediction accuracy, we compare these ranked tags with the body of the test question, add weights to each tag which appears in question body because they are more likely to be chosen as tags. Then we rank the weighted tags again, and keep the top 6 as recommendations for customers to choose from.

As for the accuracy of our recommendation, we calculate an error rate using a measurement that our recommendation is right as long as one of the actual tag appears in recommended tags.

Chapter 4

Results

4.1 Data Analysis Results

In this part, we try to analyze the differences between R and Python questions in terms of popularity, hot topics, response time, and the relationship between scores of the question and its answers.

- **R & Python Questions Volumes per year**

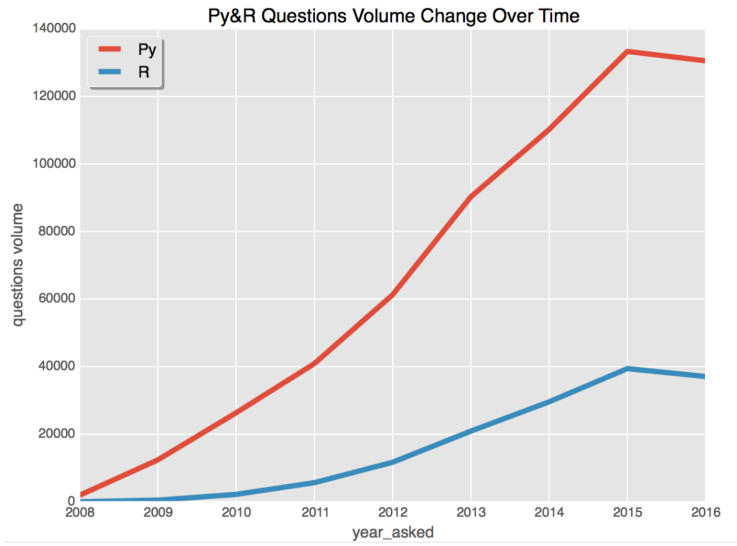

Figure 4.1: R & Python Questions Volumes per year

The amount of questions of R and Python changes in a similar pattern from year 2008 to 2016. In the first eight years, questions for both grew dramatically while

slight decreased in 2016. Despite the similar pattern, Python questions become more popular since it has a significantly huger incremental. Python questions are asked more than 130k in 2016, which is more than three times as large as that of R questions.

- **Hot Topics Comparison**

Figure 4.2: Hot Topics Comparison

We visualized the hottest topics in two languages by word clouds. The word clouds show those hottest topics with their size corresponding to their frequency of being asked. For R questions, ggplot2, dataframe, and shiny are among the hottest topics, which are mainly about data visualization and data structures. While for Python, it's Django, Numpy, Pandas, and Matplotlib that are among the hottest, which are mainly about website framework and data structures. Obviously, Python questions have more applicable fields other than data related ones. Meanwhile, both questions show a popularity concerning data manipulation.

- **Response Time Comparison**

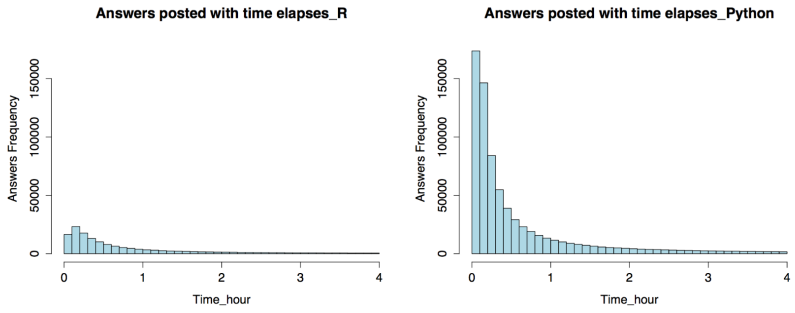

Figure 4.3: Response Time Comparison

We also look at the time it takes to have the first answer after a question posted. The mean time is 48 days for R, while 70 days for Python. And the median time is about 45 minutes for R, while 29 minutes (converted from .02 day NEED COFIRMATIONs) for Python.

- **Robust Linear Regression**
  In our case, the robust linear regression is done by iterated re-weighted least squares (IRLS). And We use the Huber weights as the weight function. cases with a large residuals tend to be down-weighted compared to all weighted as 1 in the Ordinary Least Squares method.

```
##
## Call: rlm(formula = r_agg$Score.y ~ r_agg$Score)
## Residuals:
##        Min         1Q     Median         3Q
Max
## -555.89915   -1.01373   -0.01373    1.05344
460.77894
##
## Coefficients:
##             Value     Std. Error t value
## (Intercept)    1.0137    0.0050   202.6034
## r_agg$Score    0.9328    0.0005  1748.1825
##
## Residual standard error: 1.503 on 124738 degrees
of freedom
```

```
## Call: rlm(formula = python_agg$Score.y ~
python_agg$Score)
## Residuals:
##       Min        1Q    Median        3Q
Max
## -966.6626   -0.9064    0.0936    1.2001
2270.8020
##
## Coefficients:
##                   Value     Std. Error t value
## (Intercept)         0.7999    0.0026   302.0202
## python_agg$Score    1.1065    0.0001  8609.0415
##
## Residual standard error: 1.621 on 539236 degrees
of freedom
```

Figure 4.4: Robust Linear Regression

Looking into the data, we find clear positive linear relationship between the question's score and its highest rated answer's score. By fitting robust linear regression, we have two models for R and Python respectively.Based on those two models, you probably will get higher scores by answering popular questions within those two categories and by answering a Python question if choosing across two categories.

- **Answer Score EDA**
  R and python answers summary:

Min.	1st Qu.	Median	Mean	3rd Qu.	Max
-38.000	0.000	1.000	3.028	3.000	8384.000

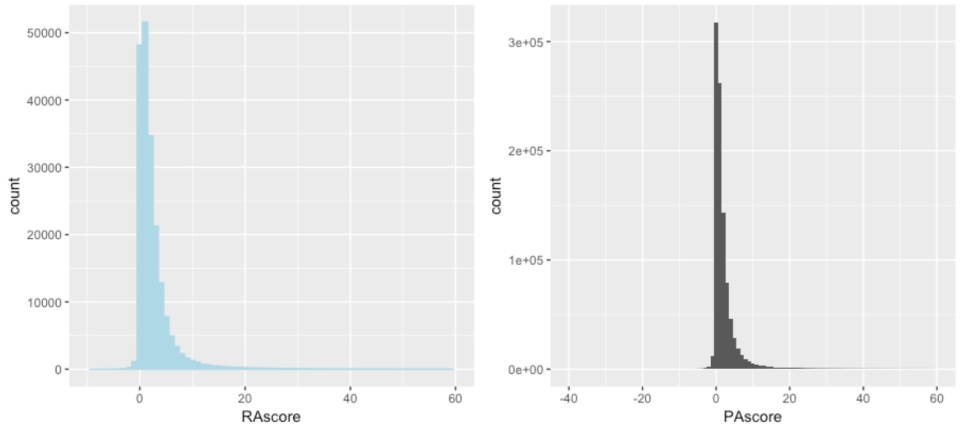

Figure 4.5: R and python answers summary

Based on scores of answers, we choose the good answer threshold = 3, the 3rd quantile.

- **Label Questions**
  Label = 1, questions get good answers, Label = 0, otherwise.

  R Questions: Total number of questions is 147071. 46435 of them get acceptable answers while 100636 of questions haven't got an acceptable answer.

  Python questions: Total number of questions is 607276.177099 of them get acceptable answers while 430177 of questions havent' got an acceptable answer.

- **Logistic Regression**
  Label prediction: Split data into train(80%) and test(20%), do logistic regression, calculate prediction accuracy.

  R Questions: Prediction Accuracy = 0.7927178;

	Estimate	Std.Error	z value	Pr(>abs(z))
(Intercept)	-1.7259480	0.009928074	-173.8452	0
Score	0.6181663	0.004818347	128.2943	0

Figure 4.6: Label prediction for R Questions

Python Questions: Prediction Accuracy = 0.7923346

	Estimate	Std.Error	z value	Pr(>abs(z))
(Intercept)	-1.7462727	0.004808411	-363.1704	0
Score	0.5649856	0.002307515	244.8459	0

Figure 4.7: Label prediction for Python Questions

- **Gradient Boosting Trees: XGBoost**
  $Score \sim tag\_count + max\_tag\_freq + sum\_tag\_freq + body\_length + title_length + body\_word\_count + title\_word\_count + code\_blocks\_count + code\_comments_count + url\_count + img\_count + code\_length + comments\_length$

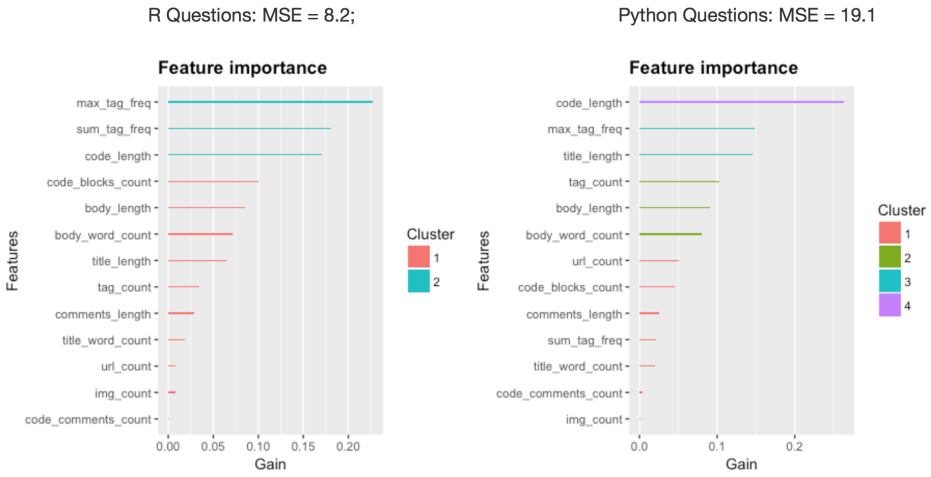

Figure 4.8: Feature importance for R&Python

4.2 Tag Recommendation System Results

We use a visualization tool (Figure 1) to visualize our topic model. To use it, click a circle in the left panel to select a topic, and the bar chart in the right panel will display the 30 most relevant terms for the selected topic, where the definition of relevance of a term to a topic is

$$log(p(term|topic)) + (1-\lambda)log(p(term|opic)/p(term))$$
for a given weight parameter, $0 \leq \lambda \leq 1$

The red bars represent the frequency of a term in the chosen topic (proportional to $p(term|topic)$), and the blue bars represent a term's frequency across the entire corpus (proportional to $p(term)$). The area of the circles depends on each topic's overall prevalence.
Change the value of λ to adjust the term rankings – small values of λ (near 0) highlight potentially rare, but exclusive terms for the selected topic, and large values of λ (near 1) highlight frequent, but not necessarily exclusive, terms for the selected topic.

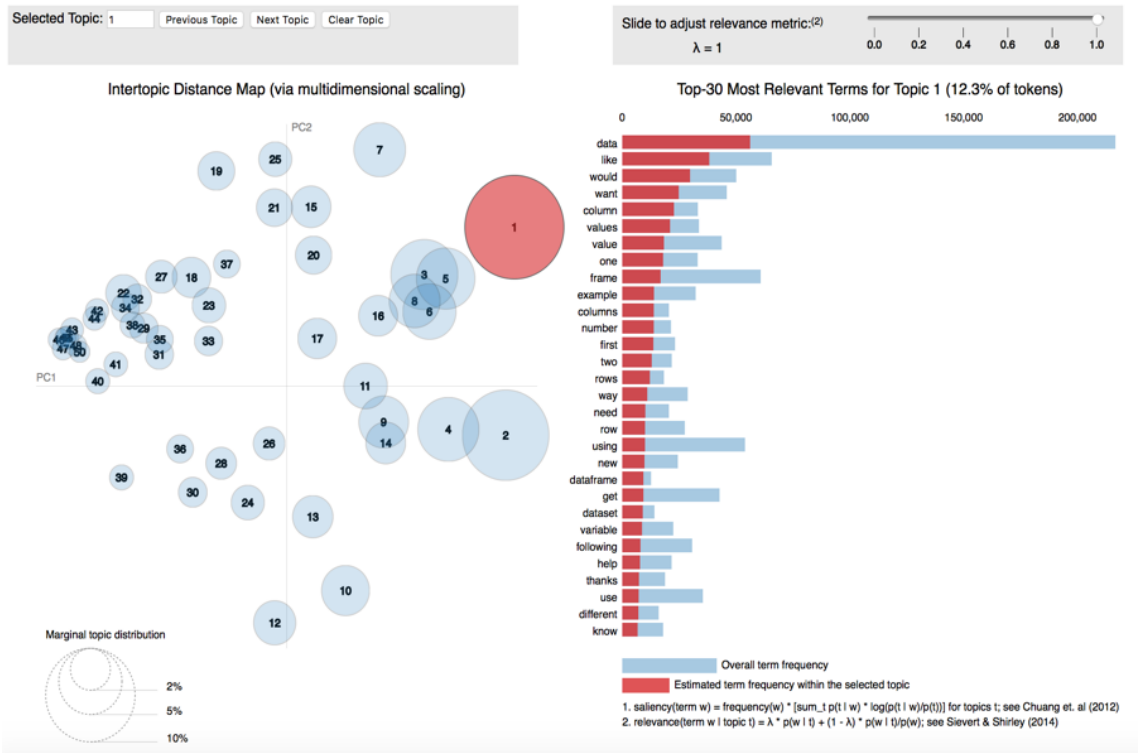

Figure 4.9: visualization and interpretation of topics

Remark: The layout of LDAvis, with the global topics (50 topics in our case) view on the left, and the term barcharts (with topic 1 selected) on the right. Linked selections allow

users to reveal aspects of the topic-term relationships compactly.

We run our algorithms and predict tags for the test set. There are some sample of our recommendation results listed in table below.

Body	Actual Tags	Recommend Tags
[struggling, developing, bubble, chart, plotly...	[plotly]	[plotly, plot, size, python, shiny, googlevis]
[plotting, stacked, bar, graph, use, geom, tex...	[ggplot2, geom-text]	[ggplot2, graph, geom-bar, geom-text, datafram...
[two, data, frame, df, df, ncol, nrow, col, co...	[dataframe]	[dataframe, row, plyr, list, for-loop, vector]
[two, data, frames, dput, data, frames, given,...	[search, data-manipulation]	[time, subset, date, matching, match, coordina...
[time, series, climate, data, years, base, plo...	[ggplot2, time-series, zoo]	[plot, ggplot2, zoo, line, time-series, smooth...
[trying, modify, values, column, rows, specifi...	[dplyr]	[dplyr, dataframe, range, split, tidyr, string]
[struggling, hours, get, match, replace, gsub,...	[regex, gsub]	[regex, gsub, string, stringr, text, quotes]
[want, make, curved, text, around, ggplot, coo...	[dataframe, ggplot2]	[ggplot2, plot, bar-chart, geom-bar, dataframe...
[trouble, functions, tried, vectorize, functio...	[integrate]	[function, integrate, integration, package, pl...
[simple, dataframe, two, vectors, speed, id, l...	[subset]	[dataframe, apply, subset, rows, function, con...

Figure 4.10: A sample of comparison between actual tags and recommended tags

After prediction, we calculate an error rate as a measure of goodness of our model and the error rate is calculated using method described in Section III. The error rate is 27%, which is a pretty low error rate, since in many recommendations, although we don't recommend the exact same tags as the actual tags but the recommended tags are strongly related to the questions and actual tags.

Chapter 5

Conclusion

5.1 Conclusion of Data Analysis

To sum up, Python questions are more popular and more diverse than R questions. Thus, generally it takes less time for Python questions to be answered. In both categories, there is a positive relationship between between the question's score and its highest rated answer's score. And it tends to be have a higher score for the answer of a more popular question. We are confident that whether a question is solved or not depends on the question's score, and as we built a model to predict question scores, we can infer that some text features are relevant to the question's outcome. Our suggestion is to try to tag your questions with some popular and regularized tags and control your codes' length and title's length short included in your question. Popular tags and brief description will make you a stack Overflow star and you don't need to worry about your programming homework anymore.

5.2 Conclusion of Topic Modelling

In short, we use LDA, k-NN algorithms and a weighting technique to do tags recommendation, which is very useful in real world. Now, all tags for questions on Stack Overflow are added by customers themselves, Stack Overflow only recommends tags after customers input some letters of the tags they are adding. But what we do in this project can help add tags on questions automatically after the questions are finished. It's very helpful for all customers, especially for customers who don't know how to add tags very well. And our recommendations are usually right and suitable due to the low error rate.
Further research should be directed towards the application of our model, and improving the accuracy of our recommendation by correcting details in our algorithms, such as, improving our weighting technique and learning more about the objects which we add weights on.

Appendix A

Reference

Here is the reference we used:

1.Ralf Krestel, Peter Fankhauser, Wolfgang Nejdl. Latent dirichlet allocation for tag recommendation, Proceedings of the third ACM conference on Recommender systems, October 23-25, 2009.

2.Yang Song , Ziming Zhuang , Huajing Li , Qiankun Zhao , Jia Li , Wang-Chien Lee , C. Lee Gile. Real-time automatic tag recommendation, Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, July 20-24, 2008.

3.Ralf Krestel, Peter Fankhauser. Tag Recommendation using Probabilistic Topic Models, 2009.

4.Nikolas Landia, Sarabjot Singh Anand. Personalised Tag Recommendation, 2009.