Why, When, and What: Analyzing Stack Overflow Questions by Topic, Type, and Code

Writers:

Miltiadis Allamanis, Charles Sutton
School of Informatics, University of Edinburgh, Edinburgh EH8 9AB, UK

Published in: Mining Software Repositories (MSR), 2013 10th IEEE Working Conference

Implemented by:

Md Mainuddin

Date: 05/02/2018

1. Introduction

Programmers seek answers on different issues on Stack Overflow. Analyzing the questions and answers we can get an insight about what aspects of programming and APIs are most difficult to understand. In this paper the researchers presented a topic model by which they created question concepts and question types. Then they connected the question concepts and types to find which question concepts have what type of questions.

In question concept model, the questions are categorized into different concepts such as “database” or “build”. By these concepts we can understand which concepts are confusing. Then questions are categorized into different types like “how to build” or “how to run”. From this model we can understand what type of information users are seeking for a concept. In the next step, authors connected question concepts and types to perform analysis like “What types of questions are most commonly asked about the *Date* object in Java?” In the final analysis, we will find the relationship between *Date* and question types *conversion* and *formatting*.

2. Methodology

The MSR challenge database for Stack Overflow data is available as PostgreSQL dump. I restored the dump into a PostgreSQL database to perform the queries. The input data for LDA modeling was extracted from the database and text documents are created using python. The text documents then used as input data to create the LDA models for further analysis. LDA modeling scripts are available from class lecture which is used with slight modifications. SQL queries are included in respective sections.

Work steps

1. Select working dataset from Stack Overflow database. We are interested on Title and body for our analysis
    a. Build queries to select the questions of the selected categories
    b. For each question select their answers
2. Extract text from the questions and answers, save in files for further analysis. Separately save the code section from answer for our code analysis.
3. Perform LDA using the documents saved from the questions and answers in previous step. This is our **Question Concepts** model.
4. Include both text and code and perform LDA on the dataset to create our **Code & Text** model.
5. Using the text documents, perform some NLP with POS tagging on data to keep verb phrases on the LDA model. This will end up with the **Question Types** model. This task is not implemented
6. Analyze how different topics are assigned to the same document. Study the covariance of the topic assignments to figure out *what are the confusing types and topics of questions*. This is the outcome of the connection between the question concepts and types. This task is not implemented.
7. Select number of questions asked in different programming languages and group by days of week to figure out which days are busy and which are slower.

3. Question Topic Models

For this implementation, a subset of questions is selected from the stackoverflow dump. Some selected categories of questions are chosen for the experiments. For programming languages, Java and Python is selected; some filtered tags are selected for Android, CSS, SQL, Version Control, and Build Tools categories. The categories and their respective number of questions found in the database are provided in the following table:

Table 1: Categories of questions used

Category	Count	Related Stack Overflow tags
Java	281508	java, java-ee
Python	122708	python
Android	210799	android
CSS	86961	css, css3
SQL	218326	sql, mysql, sql-server, postgresql, databases
Version Control	33314	git, mercurial, svn
Build Tools	12990	ant, maven

SQL query used for this data:

```
SELECT COUNT(*) FROM posts WHERE post_type_id = 1 AND (tags LIKE '%<java-ee>%' or tags LIKE
'%<java>%')
```

This query stands for all other tags.

3.1 Question Concepts

Procedure:

First, we need to create the question concepts model for our analysis. For this task, we take all the questions containing the tags mentioned in the table 1. We select the question titles and body from each question and their answers, extract the text and save into files to use as our source documents of our LDA modeling. The R script provided in the class is used for the modeling and found the question concepts presented in table 2.

Table 2: Sample question concepts

Top Words	
A1	maven, project, ant, use, file, build, plugin
A2	java, can, use, method, class, code, need
A3	python, use, like, code, file, get, function
A4	git, branch, repositori, file, chang, use, work
A5	use, object, get, code, file, can, like
A6	chang, creat, use, like, project, will, code
A7	use, code, text, class, html, problem, need
A8	use, can, python, need, object, list, will
A9	css, div, page, like, can, html, use
A10	can, use, python, like, want, work, code
A11	use, work, can, control, make, css, look
A12	queri, tabl, sql, databas, need, like, can
A13	element, will, use, css, just, class, can
A14	use, tabl, queri, mysql, want, sql, data
A15	python, will, run, use, tri, code, file
A16	can, need, one, python, want, will, first

Findings:

The stemmer trims the words having **s** or **es** at the end, but we can understand what those words are. By this presentation of concepts, we get an idea about which concepts are confusing but still we cannot figure out why these are confusing or what people are trying to achieve on these concepts. From this table we see that similar question concepts are clustered in groups. For example, we have some build tools related concepts (A1), some Java related topics (A2), similarly we found some topics related to databases (A12, A14) and so on. We also find some useful information about which terms are common in the concepts.

These results are not exactly similar to the results in the paper, but we get an expected clustering of the concepts. The differences come primarily from different steps of preprocessing like stemming, tokenizing, the corpus and also for the variation of iterations and number of topics determined for the LDA model.

3.2 Code & Text Model

Procedure:

Many questions contain code snippets in addition to text. For this model, we have uses the text from question title and body as well as the code embedded into the body section of the questions (and the answers of those questions). When extracting text from the question body, we saved the code section into separate file and for the LDA model, we used the code document along with the text documents. The topics discovered by this model is presented in table 3.

Table 3: Sample code and text Java topics of questions

Top Words*	
C1	java, new, list, string, name, *void*, creat
C2	use, can, class, object, get, need, want
C3	class, new, use, java, file, object, list
C4	method, will, need, new, file, object, code
C5	*java*, *string*, *new*, *org*, *public*, *name*, *privat*
C6	java, class, new, can, tri, thread, object
C7	*new*, *java*, *public*, *class*, *int*, *static*, *org*
C8	method, thread, class, use, new, can, need
C9	name, public, int, android, class, system, return
C10	java, can, use, method, need, string, xml
C11	string, use, can, file, will, name, java
C12	java, *string*, *new*, *public*, list, add, *org*
C13	class, method, file, need, use, code, string
C14	use, thread, method, object, need, client, xml
C15	can, java, file, will, use, code, method
C16	use, java, will, need, work, code, get
C17	method, java, file, need, thread, use, can
*Words in *italic* are code identifiers	

Findings:

From the table we find that many topics contain code identifiers only while text only and mixed topics exist as well. In many questions users mention a code snipped and ask the mentors to fix an error on the code. Sometimes users ask for implementation of some concepts or algorithms, so code snippets are included in the answers of those questions.

As described in the previous section, these results are not similar to the original results as well. However, the results are very reasonable and the causes of dissimilarity is same as above.

3.3 Hobby or Serious Work?

Procedure:

For this task, we examine some programming language usage on a daily basis.

We select the questions count for different programming languages and grouped them by days of week. Here we calculated percentage for each language for the days. The result is presented in table 4.

Table 4: Percent (%) of questions asked on a specific day for various tags

	Java	Java EE	Android	JDBC	Python	C#	RoR	SQL Server	C++	Maven	.NET	iPhone	XML	All
Mon	15.9	16.5	16.1	15.5	15.2	16.0	15.5	16.3	15.3	15.9	16.0	16.1	16.0	15.6
Tue	17.3	18.0	17.1	18.8	16.7	18.0	17.0	18.6	16.5	18.8	18.0	17.5	18.0	17.4
Wed	17.5	18.6	17.2	17.5	17.0	18.1	16.8	19.1	16.6	17.8	18.6	17.4	18.3	17.6
Thu	17.2	17.2	17.0	18.0	16.5	17.9	16.5	18.9	16.7	18.8	18.2	16.9	18.0	17.4
Fri	15.3	15.3	15.2	14.8	14.9	15.7	15.0	16.0	15.0	16.1	16.0	15.3	15.4	15.7
Sat	8.3	7.5	9.0	7.5	9.7	7.2	9.5	5.6	9.8	6.3	6.6	8.8	7.0	8.3
Sun	8.5	7.0	8.3	7.9	9.9	7.1	9.8	5.5	10.1	6.4	6.6	8.1	7.2	8.1

Findings:

From the above table we find that weekends are less busy days on Stack Overflow for all languages. Among weekdays, Mondays and Fridays are somewhat relaxed or slower days. The remaining days (Tue, Wed, Thu) are the busiest days.

4. Conclusion

From the topic model we can get a picture of what type of problems do the people ask on stack overflow and what particular topics are mostly asked for a problem area.

In the topic models implementation, the results are slightly different from the paper results but we clearly found clusters on different concepts. The difference is primarily because of data preprocessing, where questions and answers are extracted from the database raw data. Additionally, there are lots of possibilities to get different data from the stemming, tokening, and other steps of LDA modeling. Also LDA modeling results vary a lot according to the number of iterations and number of topics defined for the modeling.

However, the results which are solely dependent on the SQL queries are exactly same as the original results. The first and last tables are example of such results.

References:

1. A. Bacchelli, "Mining Challenge 2013: Stack Overflow," in *The 10th Working Conference on Mining Software Repositories*, 2013, p. to appear.
2. D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation," *the Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003.
3. A. K. McCallum, "Mallet: A machine learning for language toolkit," 2002.
4. M. F. Porter, "Snowball: A language for stemming algorithms," 2001.
5. M. Allamanis and C. Sutton, "Mining Source Code Repositories at Massive Scale using Language Modeling," in *The 10th Working Conference on Mining Software Repositories*, 2013, p. to appear.
6. IR Lab scripts from this course canvas.