Automated Stack Overflow Question Tagger

Authors
Kim, Jawewoo / 20170149
Kim, Taeyoon / 20150860
Lee, Jun Hyeong / 20150608
Jun, Seonyoung / 20140507

Abstract

Stack Overflow is a popular platform for programmers to share knowledge and insights in programming. Each question posted on the website has one or more tags attached to it to help users identify the topic of the question. However, sometimes choosing a tag for the question can be cumbersome or even difficult. We propose a tool that suggests an appropriate tag for the question in favor of the author. We introduce two possible models for finding tags and compare their performances. Furthermore, we provide a solution that merges two models in the hopes of improving the accuracy of auto tagging the questions.

1 Technical Approach

1.1 Gathering Data

We first collected the data to be used as the corpus from Stack Overflow using its API. We wrote a script that crawls through the website starting from the most recent posts. Due to the limitations on the quota, we ran the script for a few hours and collected around 8000 posts. The data contained the title, the contents, the URL etc., and most importantly, the tags attached.

1.2 Finding Important Sentences

We originally planned to use the result of 'finding important words', but we later decided not to use any other preprocessed information to derive the importance of the sentences. Topic sentences are usually positioned at the very beginning of the paragraph in English composition. We hypothesized selecting the first sentence of the given question paragraph would represent the topic of the text. Longer sentences tend to contain more important words and detailed explanations. To get a more precise result, we decided to use the number of essential words (excluding stop words) rather than the number of all words in the sentence. After POS-tagging the target sentences, we chose only adjectives and nouns to count the length of the sentences.

1.3 Finding Important Words

We aimed to find words that contain meaningful information in the given question. Finding meaningful words in the document can be divided into two parts: TF-IDF and word depth. TF - IDF stands for 'term frequency and inverse document frequency'. Term frequency counts how frequently each word appears in every document, while IDF analyzes how unique each word is in the document. The two statistics are combined together to derive the importance of each word. For example, stop words like 'the' might be high in term frequency, but it's low in inverse document frequency since it appears frequently in all documents. But terms like 'Japan' might appear a lot in some documents but not appear in other documents. Therefore, the term 'Japan' is considered important rather than 'the'. Additionally, to analyze the importance of each word inside the sentence, We implemented a simple grammar to parse the sentence and analyzed the parsed trees to find depth of each word. We assumed that a word with more depth be related to many other words in the sentence, making it relatively meaningful. Therefore, we put higher scores for words with the most depths. Combining the two methods, we built a program that scores the importance of words in the document.

2 Evaluation

The evaluation of the predicted tags is done in two steps. First, we compared the predicted tags with the actual tags attached by the users on Stack Overflow. Then, we identified the tags that were predicted but not included in the actual tags, and examined if they should be added as valid suggestions.

2.1 Comparing with the Actual Tags

This is done with two boolean parameter options: STRICT (false by default) and INCLUSIVE (true by default). If STRICT is set to True, the model will search for tag pairs that completely match. Otherwise, the model will search for 'lenient matches', which means that it will discard any special characters and numbers in the tags. 'python' and 'python3' is an example of a lenient match. It will also match tags that are different part-of-speech forms of the same word, such as 'learn' and 'learning'. One could argue that 'python3' is a more specific tag than 'python', and one could argue that they are close enough. So the best solution here was to allow the user to choose whether to allow lenient match as an option. Some tags on Stackoverflow are not single words, but multiple words connected with hyphens, such as 'react-native'. Tags that are predicted by our model, on the other hand, always consist of one word each. With the above example, if our model predicted 'react' as a tag, it would be considered a match only if INCLUSIVE is set to True.

2.2 Checking Validity of Tag Suggestions

When the matching is complete, the model checks the validity of the predicted tags without a match. This job is very simple: they are compared with the top 1000 most popular tags on Stack Overflow, and if a predicted tag finds a match among them, it is counted as a valid suggestion. The reason behind this decision is that if the tag is used widely enough to be one of the top 1000 tags, it should be good for the post as well. These comparisons are done with the exact same method as the comparisons with the actual tags, with INCLUSIVE set to True and STRICT set to False.

3 Results

The highest accuracy for the tag prediction was achieved in the inclusive, not strict cases. The accuracy was generally lowered when we considered the important sentences. The histograms of accuracy are shown in the appendix. The average accuracy for words-only case was 0.131 while it was lowered to 0.049 for the sentence-also case. The word cloud of true positive cases are also shown in the appendix.

4 Discussion

Comparing the accuracies of words-only and sentences-also cases, considering the important sentences had no apparent benefit in finding tag words. This disproves our initial hypothesis that tag words are the important ones and will appear in important sentences such as the first or the last sentences. Using the TF-IDF method generally found seemingly important words from the text, but they were not necessarily the tag words. We believe the accuracy was not very high because tag words often do not explicitly appear in the text. Tags that were predicted correctly (true positive) were usually proper nouns such as names of products (e.g. mongodb, django, pytorch etc). Predicted tags that did not match with actual tags (false positive) were words that are used generally across various fields (e.g. duplicate, function etc). Our algorithm would be most useful if one wants to add such general tags to their question. A way to improve accuracy would be to search for some conventional features in Stack Overflow tags, such as with the hyphens. If certain words are commonly connected with other words to be used as tags, we could keep them in check, and if one of such words were identified to be a possible keyword for a post, we could find the appropriate word(s) to connect with it to form a prediction. Additionally, Stack Overflow questions can generally be categorized into a specific programming language. We could improve the accuracy of the tag suggestion by determining which language the question is about. We could have tested compiling the code snippet included in the question to determine the language. However, this feature was left out because it would be analyzing an artificial language instead of a natural one, which defeats the meaning of this project.

5 Conclusion

Although, the accuracy of our tool was not very high, we believe it was an opportunity to test our hypothesis about how people add tags to their Stack Overflow questions. We found out that tags often do not appear in the original text, and we should consider the overall meaning of the question to determine the tag. Possibly such a process would require more information than just the text of the question, such as the distribution of and the relationship between tags.

A Appendix

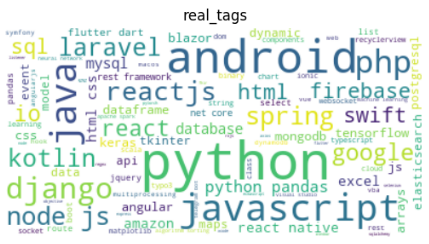

Figure 1: Word cloud of all actual tags

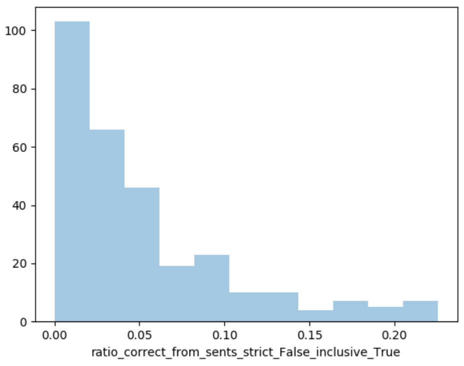

Figure 3: Prediction accuracy with both important word and sentence identification

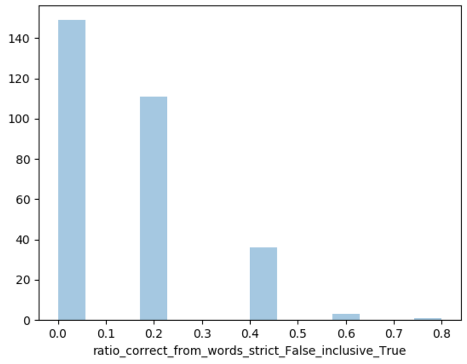

Figure 2: Prediction accuracy with the important word identification approach

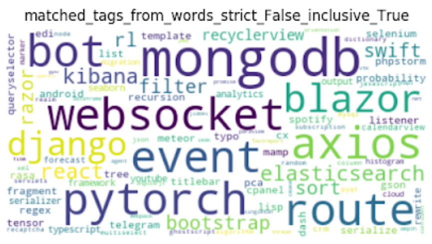

Figure 4: Word cloud of true-positive tag predictions with important word identification only

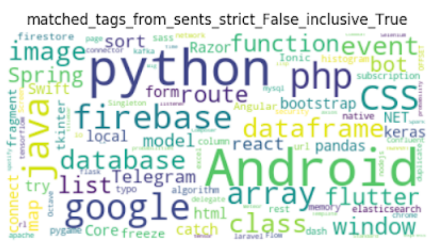

Figure 5: word cloud of true-positive tag predictions with important word and sentence identification

valid_suggestions_from_words_strict_False_inclusive_True
visual studio
studio visual
google
window
service
app
jquery selectors
sql server
android
react
web
material
asp net
runtime
django
handling
error
css selectors
api
memory management
network programming
studio code
memory
data

Figure 6: Word cloud of valid tag suggestions with important word identification only

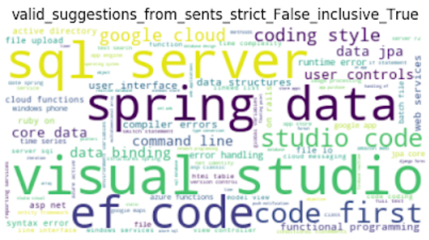

Figure 7: Word cloud of valid tag suggestions with both important word and sentence identification