Predicting the Success of Answers on Stack Overflow

Spencer Haupert, Frank Ockerman, Selena Scott, John Tsirigotis

12/17/2021

Introduction

For this project, we are exploring a dataset of all the questions and answers from Stack Overflow, focusing on R as the programming language of interest (i.e., the questions that contain an R tag) [1]. Our primary aim is to identify factors that influence an answer's score and its likelihood of acceptance.

The data include questions asked on Stack Overflow between 2008 and 2017 and are limited to questions that weren't closed or deleted. Within this dataset, there are three files available that contain information on the questions, answers, and tags specific to each post. The Tags file includes the tags used on each question besides the R tag. The Questions file includes the user ID, title, full body of text, date the question was posted, and score (the difference between the number of upvotes and downvotes). The Answers file includes the user ID, full body of text, date the answer was posted, score, and whether or not the answer was accepted (selected as the best solution to the question).

Methods

We adopted a technique known as topic modeling in order to work with the text from answers in the data. Topic modeling is a statistical method of identifying the underlying topics within a corpus (i.e., set of written texts). The process of topic modeling ultimately derives a set of K topics from the corpus, where each topic consists of a set of terms that define that topic [2]. Using Latent Dirichlet Allocation (LDA) to carry out this process, the model output consists of a topic's proportional contribution to a document (i.e., body of text of an answer), as well as the proportional contribution of a term to a topic.

To perform topic modeling, it was first necessary to create a cleaned corpus by reducing words to their root form and removing capitalization, punctuation, numbers, and common stopwords in the English language. Additionally, we only included words that were in at least 2% and at most 80% of the documents. After this process of forming a cleaned corpus, there were some documents that were left without any words, and these documents were necessarily removed prior to inputting the document-term matrix into the LDA model.

The LDA model also requires that the number of K topics is specified. Traditionally, an arbitrary value for the number of topics within a corpus is K = 20. There is also a procedure of identifying the ideal value of K by fitting several iterations of models with different values of K and comparing the coherence score of the topics in each model [3]. Coherence score measures the extent to which words within a topic are related to each other, meaning a high coherence score suggests that the derived topics consist of highly correlated words. This procedure of determining the value K through coherence score was computationally intensive and required the use of the department cluster.

We also manually created features for the Stack Overflow data pertaining to R (i.e., questions that were asked on Stack Overflow about R). Such features include, but are not limited to, the elapsed time between a question being asked and answered, the average score of the user that asked the question, and even if the title of a question contained any of the top 100 most popular R packages. We were also able to leverage the fact that the raw data contained HTML tags, which further provided interesting information about the content of the body of text. We used regular expressions to extract these features, such as whether text contained code (using the `<code>` tag) or boldface text (using both the `<b>` and `<strong>` tags).

For our analysis, we took the original data set and split it for testing and training (70% for testing, 30% for training). We created two separate models, one with acceptance as our outcome variable and the other

with score as the outcome variable. All missing values from the topic modeling variables were mean-imputed. Answer scores were truncated at 30, to avoid a small number of extreme outliers dominating the regression models.

For both models, we used a penalized regression approach to avoid issues with multicollinearity and improve parsimony. We performed lasso regression with parallelized 10-fold cross validation, selecting the shrinkage parameter which minimized mean squared error. We chose a logit link function for acceptance and an identity link function for score.

After this analysis, some simple diagnostics were performed, and the model was tested on the testing data (i.e., the 30% of the data that was initially split from the training data) to determine whether any predictions could be made about which features and topics correlate with higher scores and acceptance rates.

Results

As mentioned above, high coherence scores indicate a better fit of the LDA model. Figure 1 shows the coherence scores of models with different numbers of K topics, where models with over 12 topics have the highest coherence scores. However, the effort to name topics when K was set to 20, 15, or 10 by looking at the top 15 terms per topic suggested that a lower value of K was more appropriate. Therefore, we ran an LDA model with K = 8 and found that the topics were not overlapping excessively at this value of K, meaning the topics could be reasonably named based on the top 15 terms per topic.

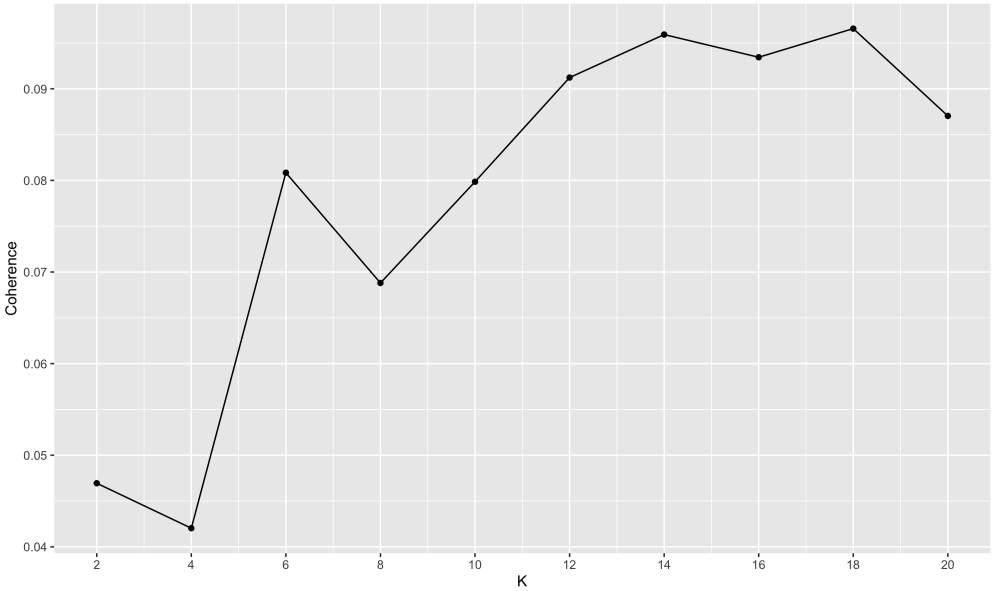

Figure 1: Best LDA Model by Coherence Score

Based on this LDA model, there were eight topics identified from the corpus of answers. Table 1 shows the names we gave those topics and the top 15 terms belonging to those topics.

Table 1: Top 15 Terms Per Topic

Topic	Terms
function definitions and calls	function code call return object argument the error result method pass defin loop oper assign
dates and times	data time group date frame start sampl year format end day count creat dat long
installing and loading packages	packag work file problem test run version read instal user check comment sourc load find
vectors	true fals vector functionx element length null sep you this here solut result fun collaps
strings	you class output if charact string match option replac remov numer convert input this or
lists and variables	list variabl set type case factor model make order level answer question dont your correct
plots and figures	plot imag line descript add label point size text here color set you fill chang
data frames and matrices	column valu row name datafram number matrix col creat index tabl select subset dataset max

Figure 2 visualizes how each answer is made up of those topics with varying proportions for a random subset of 50 answers.

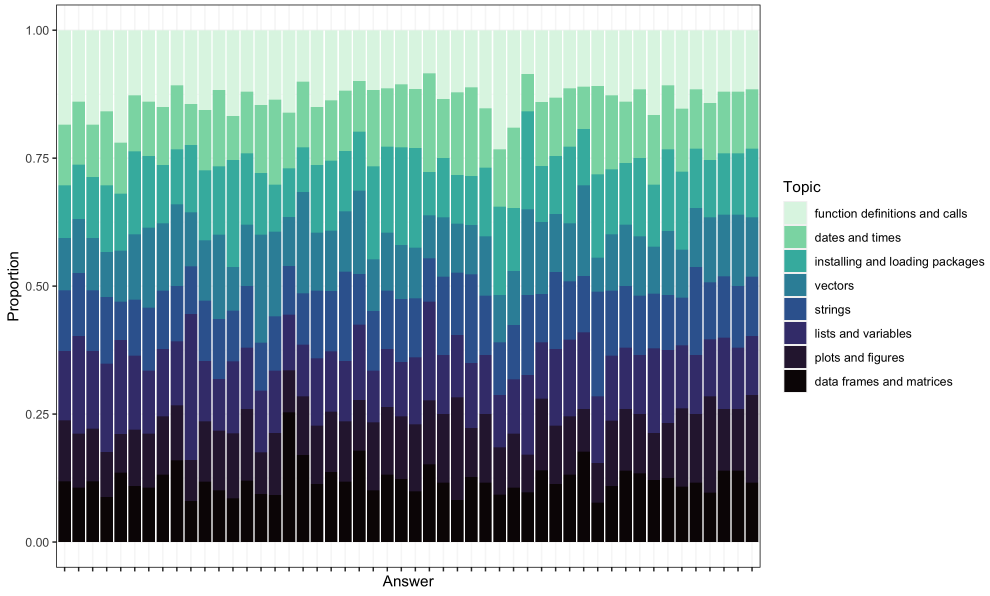

Figure 2: Topics of 50 Answers on Stack Overflow

In Figure 3, we assess our model for predicting answer acceptance with a receiver operating characteristic (ROC) curve. A good ROC curve has a sharp "elbow" with a large area below it. While our model has some small degree of predictive value, we observe no such elbow, and the ROC curve does not suggest an obvious cutoff value for classifying answers. Area under the curve (AUC) quantifies this curve and provides a simple metric for classification performance. With an AUC of 0.67, we conclude that this model fails to adequately predict which answers are accepted.

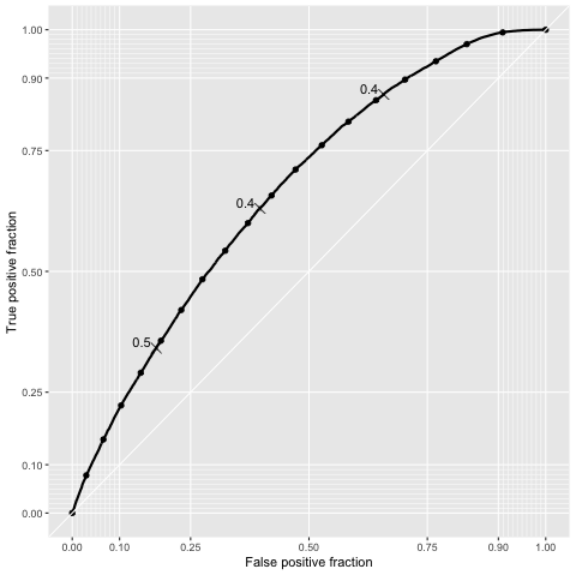

Figure 3: ROC Curve

In Figure 4, we assess our model for predicting answer score by plotting the residuals against the fitted values. Ideally, we observe no systematic effect of the fitted value on the residuals, in accordance with the assumption of homoscedasticity in linear regression. For moderate values of predicted score, we observe consistent variance in the residuals. For larger values, it is more difficult to assess the variance, although there appears to be a slight reduction in variance. Using the deviance ratio of the lasso model, we obtain the R^2 value, which indicates the proportion of variance explained by the model. The model's R^2 is 0.28, indicating that our model accounts for a substantial proportion of variance in answer scores. Another typical regression diagnostic is mean absolute error. Here, the model's mean absolute error is 2.55 points.

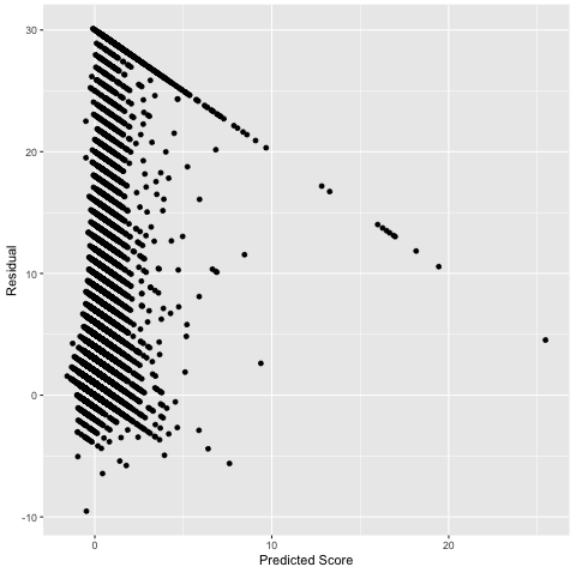

Figure 4: Residuals vs. Fitted Values

Discussion

We conclude that our topic-model based approach was unsuccessful in predicting Stack Overflow answer acceptance and score. While our overall predictive accuracy was poor, it was encouraging to see that the coefficients for topics were both significant and relatively large (compared to other coefficients), suggesting that derived topics do have some predictive value. One potential limitation to our approach was the fact that large portions of the text data were R code. Both topic modeling and sentiment analysis could have suffered from this quirk. For example, words that connote a highly negative sentiment in normal conversation (e.g., error) might not have such a negative sentiment in the context of R code. Similarly, topic modeling operates on alphabetical words and assumes that words follow some known distribution. The nature of R code could prevent meaningful topic formation because many programmers use one-off variable names and many programming operations involve non-alphabetical symbols.

While our predictive model was unfortunately unsuccessful, we were successfully able to cope with the large size of our data. The total size of the raw data was approximately 0.5 GB, presenting a considerable computational challenge. With this in mind, we applied a variety of big data techniques to make our analysis more tractable. First, we used regular expressions to efficiently extract features from the raw text. During the topic modeling step, we ran the LDA model (a computationally intensive, iterative method) on the Biostatistics cluster. Finally, during the logistic regression modeling stage, we utilized data.table for fast data manipulation, and we parallelized the logistic regression models using the *parallel* package. Because of these speed-focused choices, analyzing this large dataset was quite manageable.

This project could be extended by trying additional predictive modeling techniques, such as tree-based methods (Random Forest and XGBoost) or a Support Vector Machine. It may be the case that a model that can effectively capture nonlinear relationships, such as those mentioned above, would provide much better results. It is also true that both the topics and manually-created features constitute a major data reduction with a large loss of information. There exist deep neural network frameworks, such as convolutional neural networks (CNNs) and transformers, that are capable of taking into account a much larger portion of the information in text data, including spatial structure. We suggest future research focus on these more sophisticated techniques to leverage the rich data contained in the Stack Overflow dataset.

GitHub Repository

- https://github.com/selscott/BIOS625-Group-Project.git

Contributions

- Spencer: feature engineering, discussion, part of methods
- Selena: topic modeling, introduction, part of methods and results
- Frank: sentiment analysis, regression analysis, diagnostics, part of methods and results
- John: part of methods and discussion

References

1. Blei, D. M., & Lafferty, J. D. (2006). Dynamic topic models. In Proceedings of the 23rd International Conference on Machine Learning, 113-120.
2. "R Questions from Stack Overflow." Accessed November 17, 2021. https://kaggle.com/stackoverflow/rquestions.
3. Röder, M., Both, A., & Hinneburg, A. (2015). Exploring the space of topic coherence measures. In Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, 399-408.