PharmaSUG 2019-AD 77

An Application of Supervised Machine Learning Models on Natural Language Processing: Classifying QC Comments into Categories

Xiaoyu Tang and Todd Case

Vertex Pharmaceuticals, Boston, USA

ABSTRACT

Quality Control (QC) trackers provide text data that describe issues documented during QC of deliverables such as ADaM and SDTM datasets as well as Tables, Listings and Figures (TLFs). When documenting these issues, programmers often classify their questions into general categories, such as ADaM and Specs, SDTM and Specs, TLF, SAP, etc.. Classifying issues by programmers is subjective and time-consuming. Hence, it's important to have an objective and efficient way to classify those issues – i.e., which category the question belongs to. Having such a tool to classify free text comments not only enables us to classify those questions in just one click, but also provides us information on frequencies of each category to know the categories of most concern. Our goal is to use natural language processing (NLP) tools and machine learning methods to model such a classifier using data from the QC Tracker text. We will investigate the relationship between categories and questions, and compare several models, such as support vector classifier, random forest, and etc. to obtain the prediction accuracy using cross-validation for each model. Our goal is to choose a model that has the highest prediction accuracy and is the most valuable for future prediction of free text from QC Trackers and gain a deeper understanding and insight into that model. We used Python for our data manipulation and data analysis. Intended audiences are expected to have knowledge of statistics.

1. Introduction

Quality Control (QC) and an issue tracker are crucial parts of clinical trial data analysis. We applied several machine learning tools to process questions which exist as text data on issue trackers. Each question corresponds to a pre-classified user-defined category and point of issue (e.g., a single, specific issue). We treated it as a multi-class test classification problem with the assumption that each question is assigned to only one category and the pre-classified categories are accurate.

2. Data

In our original dataset, there are 31 categories. After combination and manipulation, there are 16 categories left (Table 1). For analysis, we only used the observations with unambiguous categories which are ADaM and Specs, SDTM and Specs, TFL, Raw, and aCRF and Specs. For those ambiguous categories that have low frequencies, we tried to predict their categories using the model trained by observations with unambiguous categories. Hence, we have 8570 observations in total for our analysis. Figure 1 shows the distribution of the six categories that we used to build our model.

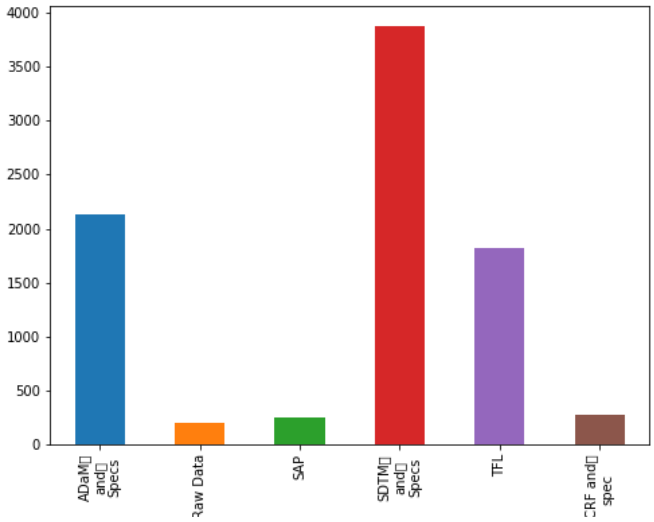
Table1:Categories manipulation

Unambiguous combined categories	Uncombined categories in original dataset
ADaM and Specs (2136)	ADaM and ADaM, ADaM, ADaM and Specs, Define ADaM

SDTM and Specs (3873)	SDTM, SDTM and Specs, Define SDTM, SDTM RAW
TFL (1824)	Figures, Listings, Tables, TFL
Raw (205)	Raw Data, Raw
aCRF and Specs (279)	aCRF and specs, aCRF
SAP (253)	SAP and TFLs, SAP shells, SAP Methods

Ambiguous categories	770-102 (26), 770-103 (16), 770-110 (20), 770-111 (40), General (10), Issues (147), Reviewer Guide (2), SDTM_ADaM (6), sheet1 (79), sheet2 (18)

Figure1: Distribution of unambiguous combined categories



3. Methods and Results

3.1 Feature Matrix

Because the questions are recorded as free text, we converted them into numerical fix-sized feature vectors which can be processed by machine learning algorithms instead of using free text with different lengths. We extracted features from those text data by tokening the questions through unigram and bigrams (one word and two word combinations), removing stop words and numbers, deleting uncommon n-grams (showing up in less than 5 documents). Then we used bag of words model and term frequency-Inverse document frequency (tf-idf) to build up feature matrix. The basic idea of bag of words model is that each question represents a bag of its words, disregarding grammar and order but keeping multiplicity.The tf–idf value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general. The calculation is that:

$$\text{Tfidf}(t,d) = (1+\log(f_{t,d})) * \log\left(\frac{N}{n_t}\right)$$

where t is a specific word, d is document d, N is total number of documents, n_t is number of documents that contain word t and $f_{t,d}$ is word t's frequency in document d. We calculated the tf-idf score for each

```
  (0, 240)      0.05439376046396702
  (0, 1866)     0.030515449344570525
  (0, 1373)     0.03778040451666607
  (0, 3019)     0.018593845780399427
```

3.2 Most correlated unigrams and bigrams

In statistics, chi-square test is used to test the independence of two events. The null hypothesis of the chi-square test is that the two events are independent. In feature selection, the two events are occurrence of the term and occurrence of the class. We ranked terms with respect to chi-square statistics. Because large chi-square statistics tend to enable us to reject the hypothesis of independence at a specific significance level, we found the corresponding unigrams and bigrams of the largest five chi-square statistics and listed those terms as most correlated unigrams and bigrams for each category. We just used chi2 in sklearn.feature_selection in python to calculate chi square statistics.

Table 2: The 5 most correlated unigrams and bigrams for each category

AdaM and Specs	Unigram	ablfl, avisit, aperiod, adsl, paramcd
	Bigram	add variables, adam spec, analysis dataset, baseline records, core variables
Raw Data	Unigram	coded, dsdrug, dsdat, randno, subject
	Bigram	subjects dm, aedecode missing, subject updated, randno subject, subject day
SAP	Unigram	Indicates, adhoc, xxxx, figure, decision
	Bigram	d figure, need d, add footnote, footnote need, annotated sap
SDTM and Specs	Unigram	raw, y, decision, updated, closed
	Bigram	column j, epoch missing, column g, usubjid vx17, raw data
TFL	Unigram	shell, footnote, y, decision, closed
	Bigram	dose group, min max, sap shell, shell missing, footnote shell

We can see that the most corrected unigrams and bigrams are reasonable and are consistent with our expectations. Specifically, we can see that for most of them, unigrams tell us *where* the problems are and bigrams tell us *what* the problems are.

3.3 Model selection

We are aiming to choose a model with the highest prediction accuracy for further analysis. We tried the four most common models for text classification: Random Forest Classifier, Support vector machine with linear kernel, Multinomial naïve Bayes Classifier and Multinomial Logistic Regression. In order avoid over-fitting, we used 5-fold cross validation, which means that we partition the data into 5 mutually exclusive complementary subsets. One fold (one round) of cross-validation performs the analysis on one subset and validates the analysis on the other subset. An accuracy of prediction can be obtained by one round of cross-validation, so 5-fold cross validation will provide us with 5 accuracies. The cross-validation accuracy for each model will be the average of the 5 accuracies in those 5 rounds. We compared the cross-validation accuracies and subsequently calculated the prediction accuracies for each model by importing functions from sklearn in python.

Sample code to get cross-validation accuracies:

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC
from sklearn.model_selection import cross_val_score
```

```
models = [
    RandomForestClassifier(n_estimators=200, max_depth=3, random_state=0),
    LinearSVC(),
    MultinomialNB(),
    LogisticRegression(random_state=0),
]
for model in models:
  model_name = model.__class__.__name__
  accuracies = cross_val_score(model, features, labels, scoring='accurac
y', cv=CV)
```

```
cv_df.groupby('model_name').accuracy.mean()
```

We chose the linear support vector classifier for further analysis because that is the model with the highest cross-validation accuracy (Figure 2 and Table3).

Figure 2: Cross-validation accuracies for each model.

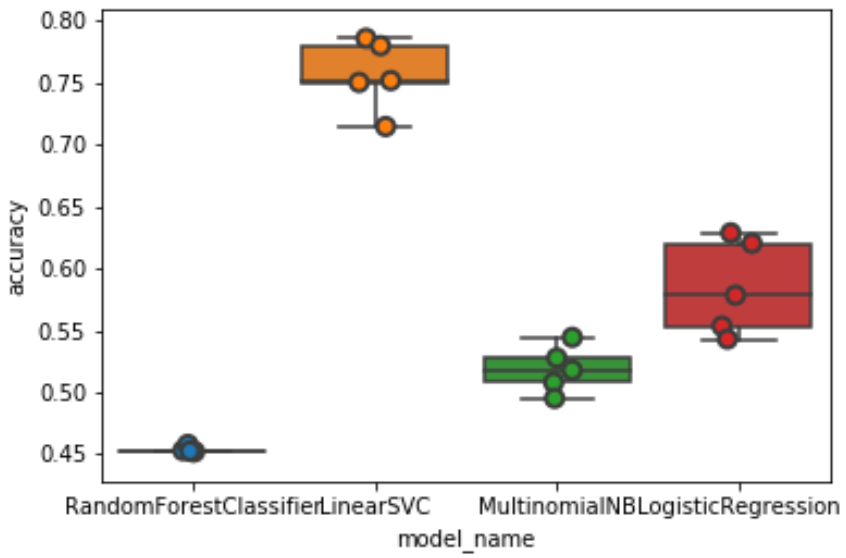


Table 3: Cross-validation accuracies for each model.

Model Name	Accuracy
Linear support vector classifier	0.756685
Logistic Regression	0.584799
Multinomial NB	0.518537
Random Forest Classifier	0.453325

3.3.1 Linear support vector machine

A linear classifier on p features has a decision boundary expressible as p-dimensional hyper plane. Hyper planes can divide the covariate space into *n* parts, which *n* is equal to number of classes. The margin of a hyper plane is the smallest distance from any training observation to the hyper plane. The maximal margin hyper plane is the separating hyper plane with the largest margin. We are aiming to find a separating maximal margin hyper plane that can perfectly divide the classes. However, we cannot guarantee that we can perfectly divide those observations into classes, so we need to seek a soft margin that does the best and, at almost separating the classes, means that we allow some observations on the wrong side of the hyper plane. We will maximize the soft margin by constraining the number of observations on the wrong side of hyper plane. Observations directly on the maximal margin hyper plane or on the wrong side of the maximal margin hyper plane are support vectors. A support vector classifier is based only on a small subset of the training observations which are support vectors. In addition, we used the linear support vector classifier. Thus, the decision boundaries are constrained to be linear.

3.4 Prediction evaluation

We first split our data into training set (2/3 of the data) and validation set (1/3 of the data). We used the training set to train the model and employed the validation dataset to test how our model performs. Firstly, we fitted our training data to a linear SVC. Then, we used the model to predict categories for questions in the validation dataset. After prediction, we created the confusion matrix and heat map (Figure3) to see the discrepancies between true categories and predicted categories. In addition, we also created a classification report (Table4), including precision, recall and f1-socre, to reflect the performance of our model. Precision= TP/(TP+FP), Recall=TP/(TP+FN) and F1-score= $\frac{2}{\frac{1}{Recall}+\frac{1}{Precision}}$ (Harmonic mean of precision and recall) where TP is true positive, FP is false positive, FN is false negative and TN is true negative. Higher f1-score represents better performance of the model. The classification report can be generated automatically by using metrics in sklearn in python.

Sample code to use SVC to do prediction:

```
model = LinearSVC()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

Sample code to get classification report:

```
from sklearn import metrics
print(metrics.classification_report(y_test, y_pred, target_names=df['origin
_sheet'].unique()))
```

Figure 3: Discrepancies between true categories and predicted categories

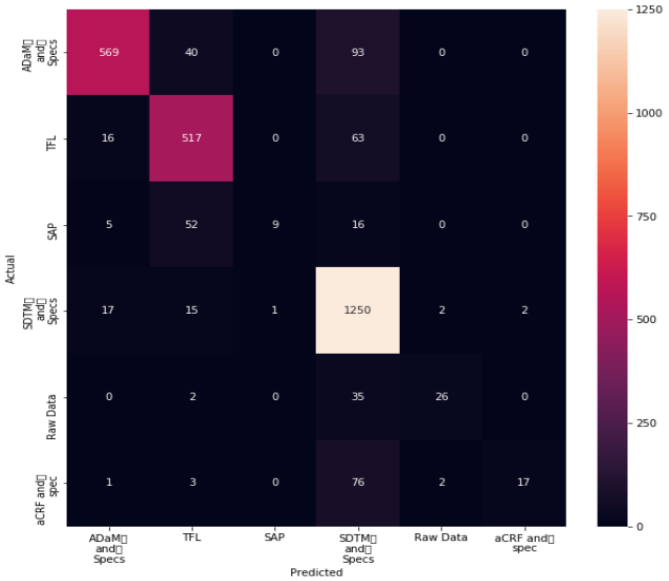


Table 4: Classification report

Category	Precision	Recall	f1-score	Counts
ADaM and Specs	0.94	0.81	0.87	702
TFL	0.82	0.87	0.84	596
SAP	0.90	0.11	0.20	82
SDTM and Specs	0.82	0.97	0.89	1287
Raw Data	0.87	0.41	0.56	63
aCRF and Specs	0.89	0.17	0.29	99

3.5 Oversampling

In our dataset, the number of questions per categories is imbalanced: the numbers of observations belonging to SAP, Raw Data and aCRF and spec are much lower than those belonging to the other classes. The problem of imbalanced data is that the overall accuracy might be high, but the recall for minority class will be low, which means that the high overall accuracy can be achieved but without providing any insights into the data. In order to deal with the imbalanced data, we used two oversampling methods: Random Sample and Synthetic Minority Over-Sampling Technique (SMOTE). The idea of oversampling by random sampling is that for each category except the majority category, SDTM and specs, we randomly chose observations from the original dataset with replacement until the observations equal to the observations of majority category.

For SMOTE, we created synthetic examples in feature space instead of resampling directly from the original dataset. First, we need to take a sample from the dataset and consider its 5 nearest neighbors in feature space. Then, we have to take the difference between the feature vector (the sample) and one of its 5 nearest neighbors which is chose randomly, then multiply this vector by a random number which is between 0 and 1 to create a synthetic data point. The created feature vector could be a new observation for that category. We repeat this process until the observations in those categories equal to the observations of the majority category. We compare the results of using the original dataset and the dataset with the two oversampling methods.

We replicated the prediction procedure by using oversampling by random sampling and SMOTE. We used RandomOverSampler and SMOTE from imblean.over_sampling in python to do oversampling. The results are the followings (Figure4/Table5 + Figure5/Table6).

Figure 4. Discrepancies between true categories and predicted categories by using oversampling by random sampling.

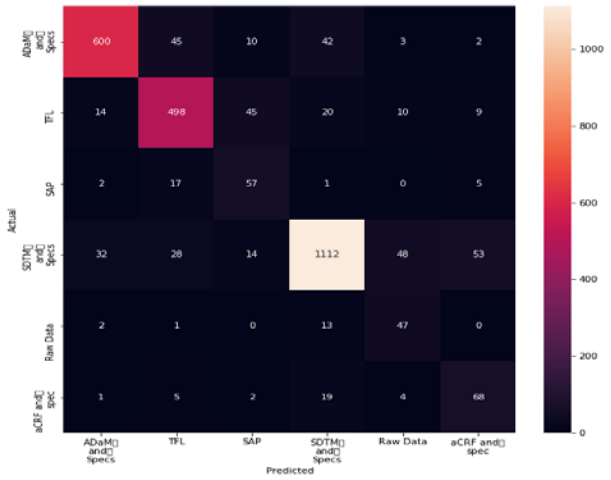


Table 5: Classification report by using oversampling by random sampling.

Category	Precision	Recall	f1-score	Counts
ADaM and Specs	0.92	0.85	0.89	702
TFL	0.84	0.84	0.84	596
SAP	0.45	0.70	0.54	82
SDTM and Specs	0.92	0.86	0.89	1287
Raw Data	0.42	0.75	0.54	63
aCRF and Specs	0.50	0.69	0.58	99

Figure 5. Discrepancies between true categories and predicted categories by using SMOTE

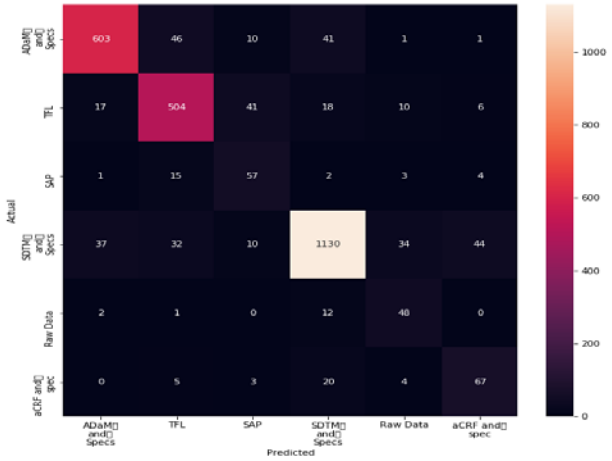


Table 6: Classification report by using SMOTE.

Category	Precision	Recall	f1-score	Counts
ADaM and Specs	0.91	0.86	0.89	702
TFL	0.84	0.85	0.84	596
SAP	0.47	0.70	0.56	82
SDTM and Specs	0.92	0.88	0.90	1287
Raw Data	0.48	0.76	0.59	63
aCRF and Specs	0.55	0.68	0.61	99

Even though the f1-scores of other categories did not change too much, the f1-scores for SAP and aCRF and Specs improved a lot. Thus, the oversampling method, indeed, enhanced the performance of our model. We can see that the model could predict most of questions correctly. For the remaining questions that were predicted incorrectly, the misclassification could be the imperfection of the model, but it could also be that the programmers misclassified those questions when they used their subjective reasoning to judge the questions. Hence, without the assumption that the classifications by programmer are totally correct, there are two possibilities for those misclassifications: 1) the predicted categories are the “true” categories, but programmers misclassified those questions. 2) Programmers classified those questions correctly and categories classified by programmers are “true” categories, but the model misclassified those questions. All the two possibilities are reasonable because the wordings of several categories are pretty similar, such as aCRF and SDTM. From the confusion matrix, among the questions in aCRF that are misclassified, most of them are predicted as SDTM. Those discrepancies could be attributed to either the model or programmers.

3.6 Prediction on data with ambiguous categories

We used the linear SVC model trained by training set to predict the data with ambiguous categories. There are 264 questions that belong to ambiguous categories.Table7 shows the prediction results.

Table 7: Prediction results for observations with ambiguous categories

Oversampling method	ADaM and Specs	TFL	SAP	SDTM and Specs	Raw Data	aCRF and Specs
Random sampling	37	191	62	38	6	30
SMOTE	34	212	52	32	4	30

From the table above, we can see that most of questions are predicted as TFL by the model. We randomly picked some examples to see if the prediction is reasonable or not. (The following cases are the questions with same predicted categories by using two oversampling methods).

Table 8: Specific examples of prediction results for observations with ambiguous categories

Questions	Predicted categories
1) We have one general updating for the ADaMs relative day. Please help confirm. 1. We use ADSL.AP01EDT as the reference date; 2. As the ADSL.AP01EDT has related day =-1 day per the SAP, the algorithm for the relative day need to be updated as: …….	ADaM and Specs
2) The column "ATM Loss of Expression" is supposed to be displayed based on SAP. However, we could not find the corresponding variable in the dataset. We will leave this column blank for now.	TFL
3) The program name of some tables in the QC plan was not correct.□ Modified as per SAP.	SAP
4) Set to the last available date among all possible raw data. " to derive DM.RFPENDTC. □Please help confirm the algorithm.	SDTM and Specs
5) Phreg will be used for Tables 11, 13, 15 with random subject effect. Shells are revised.	Raw Data
6) For XXCAT, XXSCAT, XXTEST, XXTESTCD, XXTRT, SUPPXX .QNAM,if we cannot find their information in "Vertex□Control Terminology Resource.xls", we will define them according our own understanding at this time, and using the same rules as 770-116 .	aCRF and Specs

For question (1), (2), (4) and (6), the predicted categories are obviously accurate. For problem (3), we don't believe that it belongs to any category because it's an issue with the QC plan. For question (5), it's more like TFL instead of raw data. Hence, our classifier performs fairly well. In we could have larger

sample size for the minority classes which means that if we have more features, we believe that the performance of prediction will be even better.

4. Conclusion

The purpose of this paper is to set up an efficient classifier for questions in the QC and issue trackers. We found that the support vector machine is the model with the highest prediction accuracy in terms of our data. We thoroughly utilized the model and used oversampling methods to improve our model. We concluded that our model can do a fairly well prediction of free text from QC tracker based on the data we have. Hence, our model can be a reliable reference for classifying questions into categories in issue trackers. Further research could focus on different methods of manipulating data and combining categories. In addition, more balanced data and larger sample size could be useful for accurate prediction.

References:

Susan Li. Multi-Class Text Classification Model Comparison and Selection.Towardsdatascience.

(https://towardsdatascience.com/multi-class-text-classification-model-comparison-and-selection-5eb066197568)

Susan Li. Multi-Class Text Classification with Scikit-Learn. Towardsdatascience.

(https://towardsdatascience.com/multi-class-text-classification-with-scikit-learn-12f1e60e0a9f)

Oversampling. Imbalanced-learn User Guide.

(https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over_sampling.SMOTE.html)

https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html

https://en.wikipedia.org/wiki/Bag-of-words_model

Chawla N., Bowyer K., Hall L, Kegelmeyer W (2002). “SOMTE: Synthetic Minority Over-sampling Technique”. Journal of Artificial Intelligence Research 16(2002). 321-357.