Doc2Script: A product for mapping functionality scripts to the documentations by using Multi label Classification

Prince Kumar

pk667290@gmail.com

July 2020

Abstract

We have been provided with functionality scripts and documents. The functionality scripts are in json format and documents contains html text. We approach the task by analysing functionality scripts and documents.We then extracted the html text from documents and convert it into plain text and then made classification model to map functionality scripts to documents.

We have primarily made three machine learning classifiers. Linear SVC Logistic regression Random forest It is a multilabel classification model so we have applied OnevsRest techniques.

0.1 Text Extraction

For the classification model, we need training data. we obtain our training data from the html text present in each of its sub folders. Text extraction from documents is one of the most important task. We have html text ,which we convert them into plain text using "Beautifulsoup" module of python and then make a dataframe. We also did exploratory data analysis and tried to make it more suitable for machine learning classifier algorithm to train well.

0.1.1 Documents and Functionality scripts

Our goal is to map functionality scripts to the documents. So we have functionality scripts in json format. Functionality scripts consist lot of information in which few of them are not useful in mapping ,So we just use "sstobjectlist" information.We then preprocess the text by removing all stop words and lemmatize it. The functionality script will be our data for prediction of its label.

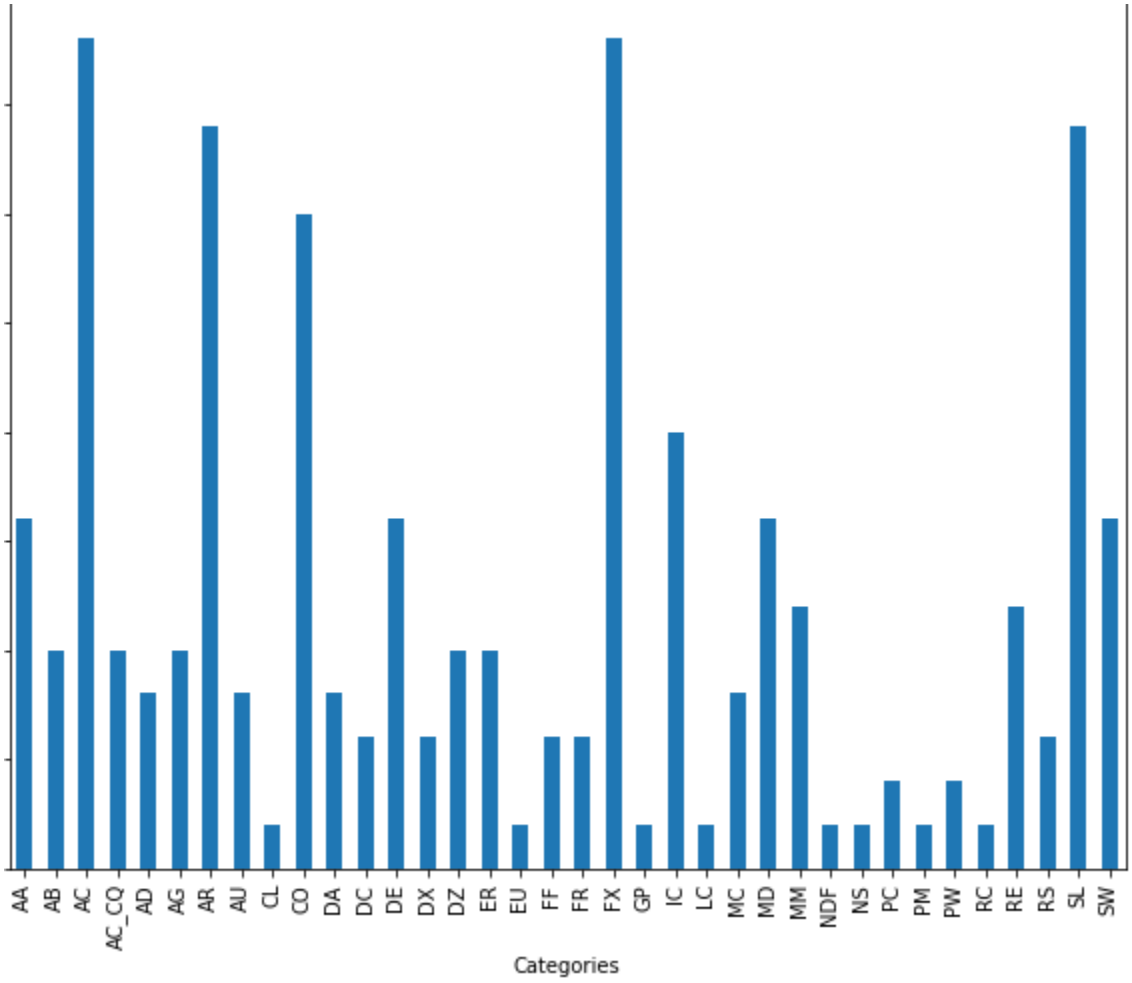
0.2 Scripts


```
{
  "rootDirectory" : "",
  "scenarioId" : "SCRPT101811001001",
  "sstObjectList" : [ {
    "scriptId" : "SCRPT101811001001-011",
    "description" : "Creation of Functionality Record For Set u
Data To be owned by Regression Team",
    "companyCode" : "US0010001",
    "scriptStatus" : "ACTIVE",
    "scriptSource" : "EN-3058108",
    "alternateRef" : "",
    "selectRoutine" : "",
    "baseRelease" : "",
    "productGroup" : "PAYMENTS",
    "productCode" : "SE",
    "scriptGroup" : "TB00if½TB00-PREUPG",
    "application" : "SEAT.FUNCTIONALITY.COMPONENT",
    "staticSetup" : "R20.ST01",
    "version" : "SEAT.FUNCTIONALITY.COMPONENT,SEAT.ZERO",
    "function" : "I",
    "transactionId" : "BASE.SETUP",
    "fieldDefinition" : [ {
      "fieldName" : "DESCRIPTION:1:1",
      "fieldValue" : "Regression Bank Base Set up",
      "fieldInput" : "YES"
    }, {
      "fieldName" : "COMPONENT:1:1",
      "fieldValue" : "RE'_'Config",
      "fieldInput" : "YES"
    }, {
      "fieldName" : "FUNCTIONALITY.REF:1:1"
```


0.3 Text clean

Text cleaning is the key step in this project. We have text in html and json format from documents and functionality scripts respectively. Conversion of html and json text in plain readable text is necessary step. Removal of stop words:- The text contains stop words so we have to remove all of them,we used "gensim" and "nltk" library from python which easily do that. Lemmatization:- We then lemmatize the text to make reduce the text.

0.4 Text distribution



0.4.1 Dataframe

Above figure represents the distribution of words in each categories in our Dataframe . This Dataframe will be our Traing data for our classification model.

0.5 2-gram

```
==> AB:
  * Most Correlated Unigrams are: compensation, nonfinancial, recipient
  * Most Correlated Bigrams are: product line, periodic charge, waive cha

==> AC:
  * Most Correlated Unigrams are: overdrawn, cutoff, bal
  * Most Correlated Bigrams are: debit transaction, credit check, availab

==> AC_CQ:
  * Most Correlated Unigrams are: stopped, stop, presented
  * Most Correlated Bigrams are: message sent, following screenshot, upda

==> AD:
  * Most Correlated Unigrams are: hierarchy, unallocated, memo
  * Most Correlated Bigrams are: attribute used, t24 account, account pro
```

0.6 Text converted into vectors

Text-processing:- we have to convert the text to vectorize it after lowering and removing stop words . TF-IDF will give us the most important words in our text

The tf-idf model is the model of tf and idf. TF:- term frequency ,it tells us the frequency of words in a given document

$$TF = Numberoftimesthetermappearsinthedoc/Totalnumberofwordsinthedoc \quad (1)$$

IDF:- Inverse-Document-Frequency,it tells us ,how rare or commom a word in the document.

$$IDF = ln(Numberofdocs/Numberdocsthetermappearsin) \quad (2)$$

CountVectorizer It provides a simple way to both tokenize a collection of text documents and build a vocabulary of known words, but also to encode new documents using that vocabulary. You can use it as follows: Create an instance of the CountVectorizer class. Call the fit() function in order to learn a vocabulary from one or more documents. Call the transform() function on one or more documents as needed to encode each as a vector. An encoded vector is returned with a length of the entire vocabulary and an integer count for the number of times each word appeared in the document.

0.7 OnevsRest

This strategy, also known as one-vs-all, is implemented in OneVsRestClassifier. The strategy consists in fitting one classifier per class. For each classifier, the class is fitted against all the other classes. Advantage of this approach is its interpretation. Since each class is represented by one and only one classifier, it is possible to gain knowledge about the class by inspecting its corresponding classifier. This is the most commonly used strategy and is a fair default choice. So we shall go by this method

0.8 Classifiers

Linear SVC :- The objective of a Linear SVC (Support Vector Classifier) is to fit to the data you provide, returning a "best fit" hyper plane that divides, or categorizes, your data. From there, after getting the hyper plane, you can then feed some features to your classifier to see what the "predicted" class is. This makes this specific algorithm rather suitable for our use.

Logistic Regression :- A contradiction appears when we declare a classifier whose name contains the term 'Regression' is being used for classification, but this is why Logistic Regression is magical: using a linear regression equation to produce discrete binary outputs . And yes, it is also categorized in

'Discriminative Models' subgroups of ML methods like Support Vector Machines and Perceptron where all use linear equations as a building block and attempts to maximize the quality of output on a training set.

Random forest :- It is an ensemble tree-based learning algorithm. The Random Forest Classifier is a set of decision trees from randomly selected subset of training set. It aggregates the votes from different decision trees to decide the final class of the test object.

0.9 Conclusion

Using various algorithms like linear SVC, randomforest classifier. we trained our model on documents text. Testing data is the text from Script Our classifier predict the labels . Thus using this multi class classification model we can map our scripts to documents.

$\mathrm{X}_train, X_test, y_train, y_test, indices_train, indices_test = train_test_split(features, labels, df.index, test_size = 0.25, random_state = 1)$

model = OneVsRestClassifier(LinearSVC()) model.fit($\mathrm{X}_train, y_train$)

$\mathrm{y}_pred = model.predict(X_test)$

print('accuracy

accuracy: 0.7358490566037735 label : ['CO']

Thus we can see that a script matched to category it belong.

0.10 Refferences

• Towards data science :susan li

• Keith Stevens, Philip Kegelmeyer, David Andrzejewski, David Buttler,proceedings of Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Jeju Island, Korea, July 2012.