Capstone Project - Machine Learning Engineer Nanodegree

Renee Jia
July.5th, 2020

QUEST Q&A Labeling
—— Improving automated understanding of complex question answer content

Part 1: Definition

1.1. Project Overview

The distinguishment of computers and human-beings is that the computers are really good at answering the questions with a fixed answer. However, humans are better at those questions which are more comprehensive and require more emotional experiences.

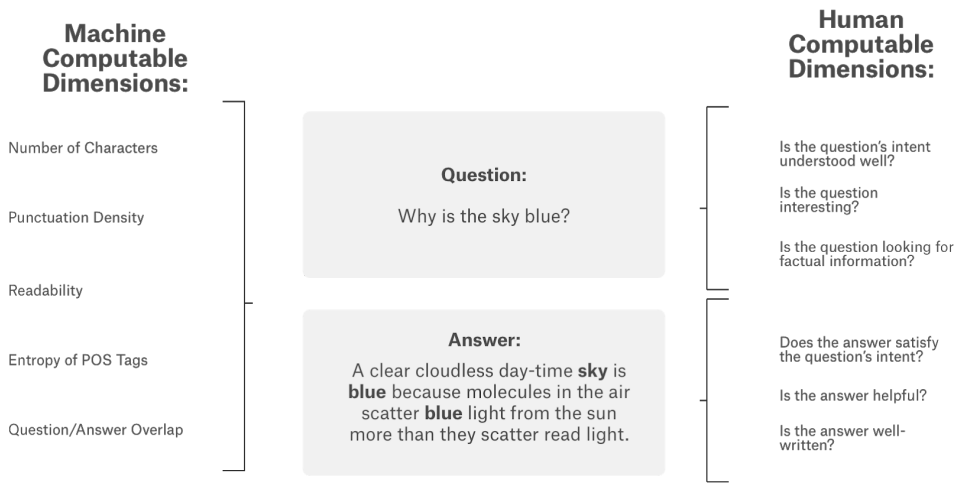
We know that machine learning could take great advantage of the super power using the computers to learn a lot of the fixed historical data and solve problems. However, due to the factor that the computers' lack of knowledge in a deeper, multidimensional understanding of context, computers may need to improve in this part.

The questions can take many forms - some have multi-sentence elaborations, others may be simple curiosity or a fully developed problem. They can have multiple intents, or seek advice and opinions. Some may be helpful and others interesting. Some are simple right or wrong.

The overall problem could be categorized as a supervised learning prediction problem: to use the dataset to build the prediction for different subjective aspects of question-answering.
The question-answer pairs were gathered from nearly 70 different websites, in a "common-sense" fashion. The raters received minimal guidance and training, and relied largely on their subjective interpretation of the prompts. As such, each prompt was crafted in the most intuitive fashion so that raters could simply use their common-sense to complete the task.

1.2 Problem statement

The chart below gives a general overview of the problem, and what the emphasis of this problem is trying to achieve in terms of bridging the gap between the human's comprehension of a subjective question and where could we take the advantage of the computer's computational power:



1.3 Problem Solving Approach

The data inputs include questions and answers from various StackExchange properties. Your task is to predict target values of 30 labels for each question-answer pair. This challenge could be considered as a prediction problem of predicting the probability of the 30 labels for each question-answer pair and calculate the probability for each of the labels.

I defined this problem as a multi-class classification problem, in order to reach a robust prediction of the classification. First and foremost, I need to download and have a glance of the overall dataset to get a sense of the dataset that I'm going to analyze. Then after the data overview, I would like to go deep to the data exploration part to see the data by category and get more sense of the dataset. After that, since this is a complex dataset, to reach better result, I'll work on data manipulation as well feature engineering.

After the pre-processing of the data, I'll start with the model building. In this part, would like to use cross validation and grid search for choosing the best parameters for model selection. Then will fit the model with the best parameter and apply the model to the testing dataset.

Below shows a specific plan to reach this problem:

1. *Retrieving the data*
2. *Data Overview*
    a. *Overview of the tables to get a sense of the datasets*
    b. *Conduct statistical analysis of the dataset and draw plots to get a sense of the files*
3. *Data Exploration*
    a. *Draw distribution plots of host, categories, target variables*
    b. *Draw distributions of the Question Title/ Body and distribution of the answers*
4. *Data preparation*

    a. *Data cleaning: Check with missing data and using techniques to manipulate data (eg: deal with null values, etc.)*
    b. *Feature Engineering*
        i. *Text based features*
        ii. *TF - IDF features exploration*
5. *Setup baseline model as benchmark*
6. *Data splitting*
    a. *Split the training dataset into testing & validation datasets*
7. *Model building*
    a. *Build different models for comparison*
8. *Define testing metrics*
9. *Model validation and testing*
    a. *Using the validation dataset to evaluate the performance of different models and compare*
    b. *Using the test dataset to generate the submission file and upload to kaggle competition*
10. *Write up*
    a. *General summary of the model comparisons*
    b. *Business insights generation*

1.3 Evaluation metrics

The model would be evaluated on the mean column-wise Spearman's correlation coefficient. The Spearman's rank correlation is computed for each target column, and the mean of these values is calculated for the overall performance of the model.

Specifically, the Spearman's correlation coefficient is calculated as:

$$r_s = \rho_{\mathrm{rg}_X, \mathrm{rg}_Y} = \frac{\mathrm{cov}(\mathrm{rg}_X, \mathrm{rg}_Y)}{\sigma_{\mathrm{rg}_X} \sigma_{\mathrm{rg}_Y}},$$

Part 2: Analysis

2.1 Data Exploration

The data includes questions and answers from various StackExchange properties. Your task is to predict target values of 30 labels for each question-answer pair.

The list of 30 target labels are the same as the column names in the `sample_submission.csv` file. Target labels with the prefix `question_` relate to the `question_title` and/or `question_body` features in the data. Target labels with the prefix `answer_` relate to the `answer` feature. The columns of the training data shows as below:

```
train_data.columns
```

```
Index(['qa_id', 'question_title', 'question_body', 'question_user_name',
       'question_user_page', 'answer', 'answer_user_name', 'answer_user_page',
       'url', 'category', 'host', 'question_asker_intent_understanding',
       'question_body_critical', 'question_conversational',
       'question_expect_short_answer', 'question_fact_seeking',
       'question_has_commonly_accepted_answer',
       'question_interestingness_others', 'question_interestingness_self',
       'question_multi_intent', 'question_not_really_a_question',
       'question_opinion_seeking', 'question_type_choice',
       'question_type_compare', 'question_type_consequence',
       'question_type_definition', 'question_type_entity',
       'question_type_instructions', 'question_type_procedure',
       'question_type_reason_explanation', 'question_type_spelling',
       'question_well_written', 'answer_helpful',
       'answer_level_of_information', 'answer_plausible', 'answer_relevance',
       'answer_satisfaction', 'answer_type_instructions',
       'answer_type_procedure', 'answer_type_reason_explanation',
       'answer_well_written'],
      dtype='object')
```

Each row contains a single question and a single answer to that question, along with additional features. The training data contains rows with some duplicated questions (but with different answers). The test data does not contain any duplicated questions.

The picture below shows the data format:

```
train_data.head()
```

	qa_id	question_title	question_body	question_user_name	question_user_page	answer	answer_user_name	
0	0	What am I losing when using extension tubes in...	After playing around with macro photography on...	ysap	https://photo.stackexchange.com/users/1024	I just got extension tubes, so here's the skin...	rfusca	https://photo.sta
1	1	What is the distinction between a city and a s...	I am trying to understand what kinds of places...	russellpierce	https://rpg.stackexchange.com/users/8774	It might be helpful to look into the definitio...	Erik Schmidt	https://rpg.sta
2	2	Maximum protusion length for through-hole comp...	I'm working on a PCB that has through-hole com...	Joe Baker	https://electronics.stackexchange.com/users/10157	Do you even need grooves? We make several pro...	Dwayne Reid	https://electronics.stac
3	3	Can an affidavit be used in Beit Din?	An affidavit, from what i understand, is basic...	Scimonster	https://judaism.stackexchange.com/users/5151	Sending an "affidavit" it is a dispute between...	Y e z	https://judaism.sta
4	5	How do you make a binary image in Photoshop?	I am trying to make a binary image. I want mor...	leigero	https://graphicdesign.stackexchange.com/users/...	Check out Image Trace in Adobe Illustrator. \n...	q2ra	https://graphicdesign.

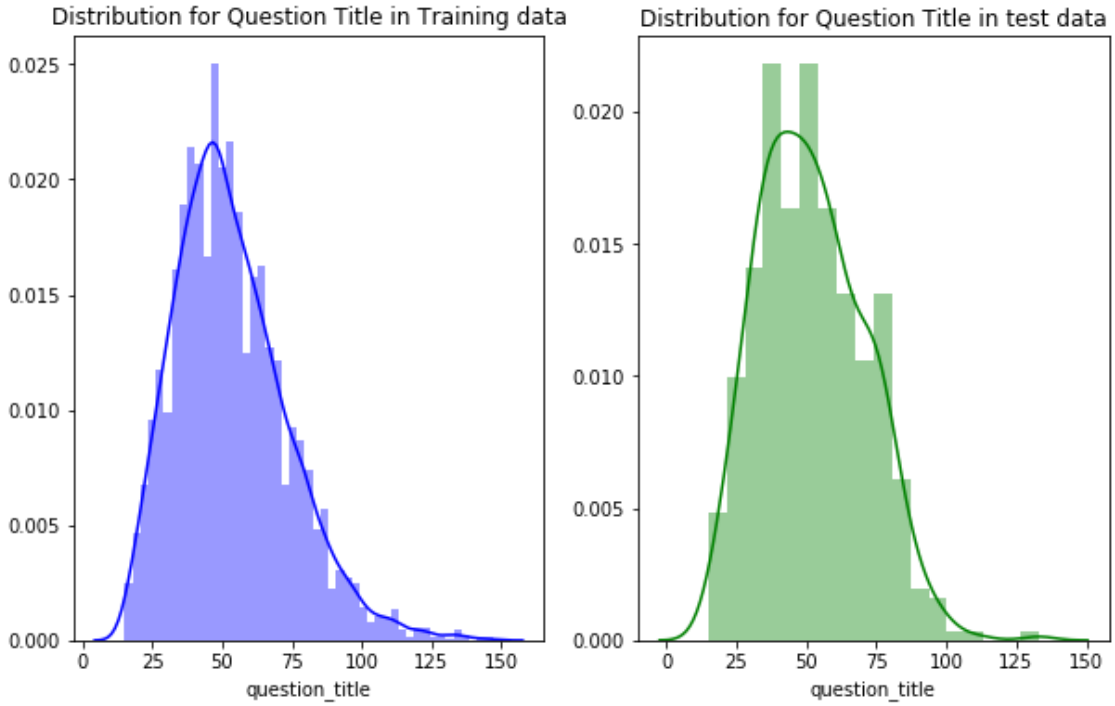
5 rows × 41 columns

2.2 Exploratory Visualization

After explore the data, below are some of the meaningful visualizations:

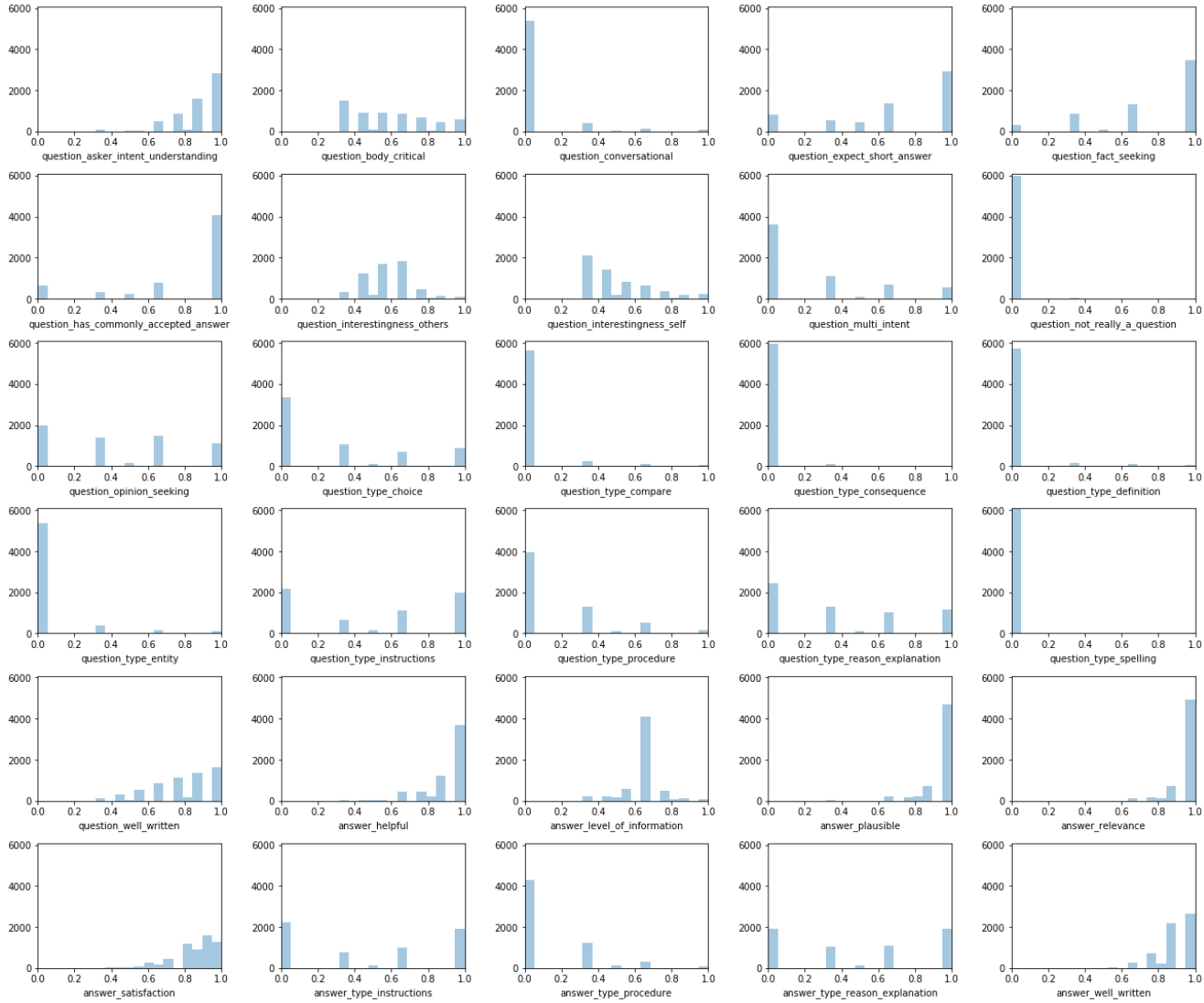
2.2.1 Distribution for Question TItle in training and testing data

As we could from the bar chart, in the dataset, the training dataset are from the question_title around 50, which contributes to the highest distribution percentage.



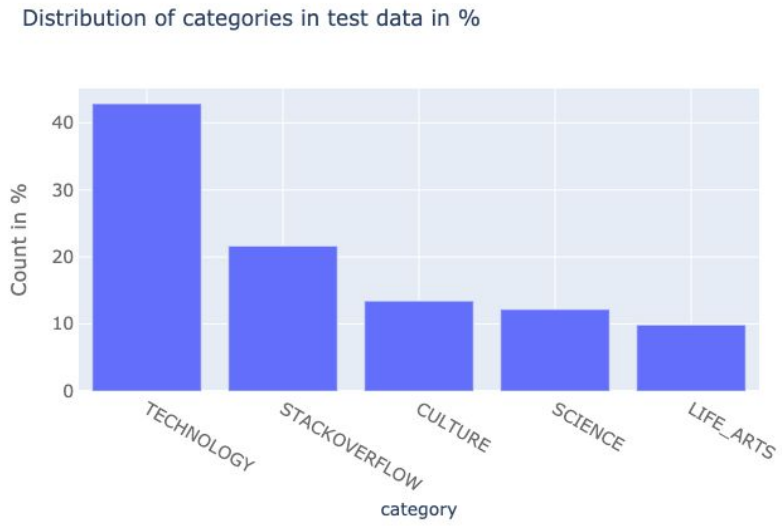
2.2.2 Distribution of target variables

Below shows the distributions of all the target variables in the training dataset:



2.2.3 Distribution of the question categories in test data

Below shows the distribution of the question categories in the test dataset, which could give us a sense of how the questions are categorised:



2.3 Algorithms and Techniques

2.3.1 Natural Language processing:

- **Stop words removal:**
  In the data processing part for large quantities of text data, this technique is widely used, since stop words occur in abundance, hence providing little to no unique information that can be used for classification or clustering.

- **TF-IDF (frequency-inverse document frequency)**
  This technique short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus
  To be more specific, the tf-idf weight is composed of two terms:
  - Normalized Term Frequency (TF): the number of times a word appears in a document, divided by the total number of words in that document; -
  - Inverse Document Frequency (IDF): computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears.
  - The final IDF(t) = log_e(Total number of documents / Number of documents with term t in it).

2.3.2 Machine Learning algorithms:

- **MultiLinearRegression Classification**
  Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. The goal of multiple linear regression (MLR) is to model the linear

relationship between the explanatory (independent) variables and response (dependent) variable. The formula runs as follows:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip} + \epsilon$$

2.4 Benchmark

The overall question could be defined as a prediction model, the benchmark model would be using the MultiOutputRegression model for setting up a decent baseline. By starting from a linear regression model, this could provide a solid baseline for further model explorations.

Part 3: Methodology

3.1 Data Preprocessing

The preprocessing of the training data consists of the followings:

1. Data cleaning for language processing
    a. Eliminate the ‘symbol’ and stopping words
    b. Using hashmap for acronyms
    c. Cleaning the text and change it to the lowercase for better machine comprehension
2. Counting the word frequency for training
    a. Program the function to analyze the word frequency in processing the question
3. Feature engineering
    a. Adding question title and body length variables as new features
    b. Adding question title and body unique words length as new features
    For the above parts, the specific new created features are as follows:

        - *Number of characters in the question_title*
        - *Number of characters in the question_body*
        - *Number of characters in the answer*
        - *Number of words in the question_title*
        - *Number of words in the question_body*
        - *Number of words in the answer*
        - *Number of unique words in the question_title*
        - *Number of unique words in the question_body*
        - *Number of unique words in the answer*

    c. Adding TF-IDF features for natural language processing
    For the above TD-IDF data processing, the specific based features are:

        - *Character Level N-Gram TF-IDF of question_title*
        - *Character Level N-Gram TF-IDF of question_body*
        - *Character Level N-Gram TF-IDF of answer*
        - *Word Level N-Gram TF-IDF of question_title*
        - *Word Level N-Gram TF-IDF of question_body*

- *Word Level N-Gram TF-IDF of answer*

3.2 Implementation

In the implementation of the model, I first split the training dataset and used the one-hot encoder to encode the different questions. Then I created the preprocessor to pre-process the data in order to fit the model building.
In terms of the model building, I choose to use the LinearRegression model for training. And I used the cross validation which set the n_split to 5.

3.3 Refinement

For choosing the best parameter, I set the grid_search in cross validation in order to choose the best parameter for fitting the model, and based on the evaluation for the spearman coefficient score, to choose the best parameters for fitting the model.

Using the grid_search, we could select the best parameters for building the model, and in the parameters setting, I choose to use the Ridge as the limit parameter for computing the loss function.

Part 4: Results

4.1 Model Evaluation and Validation

For choosing the best parameter, I set the grid_search in cross validation in order to choose the best parameter for fitting the model, and based on the evaluation for the spearman coefficient score, the score for the best fitting model would be 0.354429.

The test score for all the other splits are listed as follows:

```
  'preprocessor__title__tfidf__sublinear_tf': 1
  'preprocessor__title__tfidf__token_pattern':
  'preprocessor__title__tfidf__use_idf': True}
'split0_test_score': array([0.35226806]),
'split1_test_score': array([0.34793838]),
'split2_test_score': array([0.3568958]),
'split3_test_score': array([0.3580901]),
'split4_test_score': array([0.35695405]),
'mean_test_score': array([0.35442928]),
'std_test_score': array([0.00381215]),
'rank_test_score': array([1], dtype=int32),
'split0_train_score': array([0.52299955]),
'split1_train_score': array([0.52151352]),
'split2_train_score': array([0.51949357]),
'split3_train_score': array([0.51951444]),
'split4_train_score': array([0.52221626]),
'mean_train_score': array([0.52114747]),
'std_train_score': array([0.00142188])})
```

4.2 Justification

Using the grid search, I could choose the model with the best parameters to fit the training data and apply the built model to the test dataset and output the result.
The best parameter settings are shown as follows:

```
Pipeline(memory=None,
         steps=[('preprocessor',
                 ColumnTransformer(n_jobs=None, remainder='drop',
                                   sparse_threshold=0.3,
                                   transformer_weights=None,
                                   transformers=[('title',
                                                  Pipeline(memory=None,
                                                           steps=[('tfidf',
                                                                   TfidfVectorizer(analyzer='word',
                                                                                   binary=True,
                                                                                   decode_error='strict',
                                                                                   dtype=<class 'numpy.float64'>,
                                                                                   encoding='utf-8',
                                                                                   input='content',
                                                                                   lowercase=False,
                                                                                   max_df=0....
                                                                                             handle_missing='value',
                                                                                             handle_unknown='value',
                                                                                             mapping=None,
                                                                                             return_df=True,
                                                                                             verbose=0))],
                                                           verbose=False),
                                                  ['domain_1', 'domain_2',
                                                   'domain_3', 'category',
                                                   'is_question_no_name_user',
                                                   'is_answer_no_name_user'])],
                                   verbose=False)),
                ('estimator',
                 Ridge(alpha=20, copy_X=True, fit_intercept=True, max_iter=None,
                       normalize=False, random_state=22, solver='auto',
                       tol=0.001))],
         verbose=False)
```

Part 5: Reflections and Further Improvements

5.1 Reflections

The overall project is a very interesting one and I truly learned a lot during the whole process. And I found that in order to solve a real machine learning problem, I need to spend an incredible amount of time in the data exploration and data preprocessing to better understand the dataset.

In terms of feature engineering, for a real life dataset, there's a lot to be done with this part, especially after the data exploration, and this could add more value and more human insights to reach a better result in the overall model performance. A saying in the field of machine learning as goes :”garbage in, garbage out”. This demonstrates the importance of good data processing as well as feature engineering in the performance for real machine learning problem solving.

Also, there’re a lot of models that could be used for the same kind of problem, and how to choose an appropriate machine learning algorithm and how to do the parameter tuning are very important to reach an overall good result for the project.

5.2 Further Improvements

This is a very interesting question which could be done with a lot more tricks in the feature engineering part. Also, for the data pre-processing part, I could use more of the natural language processing techniques to further improve the model accuracy.

Moveover, in terms of the mult-iclassification problem, the MultiLinearRegression model is a very intuitive and simple method to reach the result, however, for this question, it might be better to apply more complex models such as Deep Learning or the BERT model in language processing to achieve better results.