Building Predictive Algorithms for Improving Automated Understanding of Complex Question Answer Content

Kylie Vo
kylie.vo@ischool.berkeley.edu

Abstract

Question answering (QA) has been a benchmark task in natural language understanding. However, the development of a methodology for evaluation is still challenging. One of the key issues is to find the metrics that is used for evaluation to ensure quality. This document contains my work on the Kaggle Google Quest QA Labeling competition. The challenge is to build predictive algorithms for different subjective aspects of question-answering by predicting the scores given by human raters to questions and answers on various websites. The questions and answers are scored on 30 dimensions, including whether the question or answer was well-written, relevant, helpful, satisfactory, contained clear instructions, etc. The work is intended to find a result that can hopefully foster the development of QA systems.

1 Introduction

Computers are really good at answering questions with single, verifiable answers. However, humans are often still better at answering questions about opinions, recommendations, or personal experiences (Figure 1).

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 CrowdSource team at Google Research, a group dedicated to advancing NLP and other types of ML science via crowdsourcing, has collected data on a number of these quality scoring aspects.

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.

Demonstrating these subjective labels can help to make predictions more reliable and shine a new light on this research area. Results from this will hopefully inform the way future intelligent QA systems will get built, and contributing to them becoming more human-like.

2 Backgrounds

QA systems can be found in many areas of NLP research including natural language database systems and spoken dialog systems. Therein, community QA has already attracted attention from researchers investigate information seeking behaviors and a wide range of other information-related behaviors. Evaluation of QA systems were developed as early as in 1960s (Clarke[3]). QA systems can be classified by their data sources. Some are based on structured databases containing well-formatted information (Strzalkowski et al.[5]) while others extract answers from textual documents such as newspaper corpus. A comprehensive classification of QA systems was then given by Lopez et al.[4].

Some researchers suggest that the context that rises to an information need is unique for every individual and answer that is useful to an individual in a particular context may only partially useful to others in other contexts. Currently, the evaluation of the quality of questions and answers are largely classification problems. However, the quality of an answer, or any information content, can be subjective. Giving a single predictive rating may not be sufficient. A quality assessment may depend on the relevant of the content, and among others, relevance itself is difficult to measure (Zhao et al.[6]).

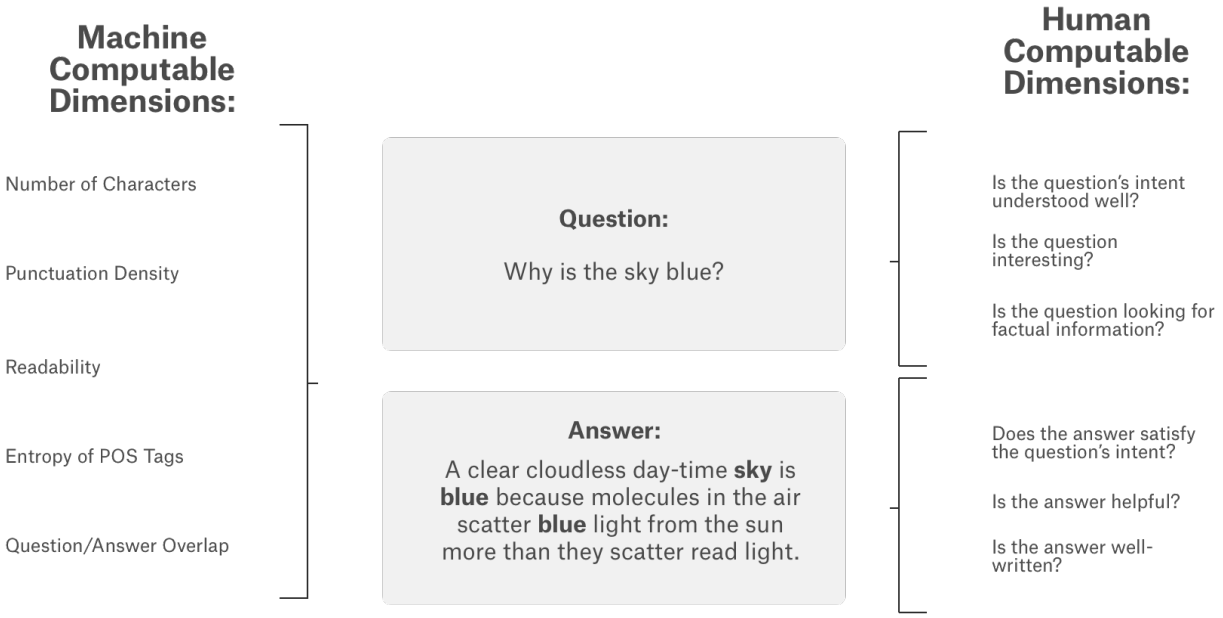

Figure 1: Machine-Human Computation Dimensions

A range of new approaches has also emerged. As a step towards a better QA metric, Chen et al.[2] suggested using BERTScore, a recently proposed metric for evaluating translation, for QA. Since it fails to provide stronger correlation with human judgements, he also proposed future work focused on tailoring a BERT-based metric to QA evaluation. Zhu et al.[7] identified a set of 13 criteria when asking human assessors to judge the quality of the information from both questions and answers. In this report, 30 criteria are used as different aspects of answer quality. Extracting necessary features is needed to construct fairly reliable models.

3 Data and Exploratory Data Analysis

3.1 The Data

The data for this competition includes questions and answers from various StackExchange properties.

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

The data includes 30 target labels for each question-answer pair. 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. Target labels are aggregated from multiple raters, and can have continuous values in the range [0,1]. Therefore, predictions would also be in this range.

3.2 Exploratory Data Analysis

The Exploratory Data Analysis is conducted on both of the train and test sets. The training set has a total of 6079 samples. The first 11 columns are the feature columns. The last 30 columns are the target variables, including the scores on both questions and answers. The test set has 476 samples and only includes the feature columns.

The following aspects of the data were examined:

1. Question category distributions
2. Repeated questions titles/bodies and most popular questions
3. Word count distribution of the question titles/bodies and the answers
4. Common words used in the question titles/bodies and the answers
5. Target variables distribution
6. Correlation between target variables
7. Distribution of Scores by Categories

Both of the train and test sets have no missing data. There are five categories in the samples: culture, technology, life arts, stackoverflow, and science. The train data set appears to have more samples on the science category, whereas the test data set appears to have more samples on the technol-

ogy category. With World Cloud visualization, it appears that common word usage distribution is similar between train and test sets.

Examine correlation in target labels, it is reasonable to see that question fact and opinion labels are anti-correlated. Answer instruction and answer reason explanation labels also appear to be anti-correlated.

It was also interesting to look at the distribution of number of words. Answer's number of words distribution is similar to question body. These can be useful for predicting target values, so I also observed the correlation of length with the target values.

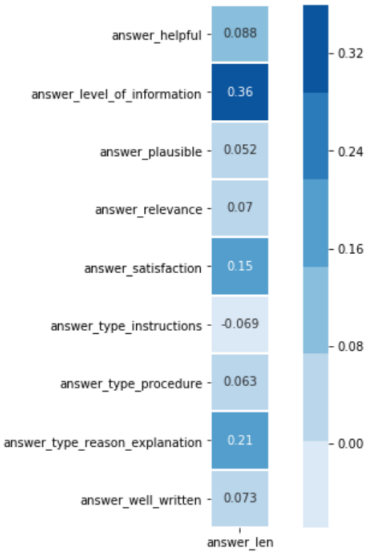

Figure 2: Answer Length Vs. Answer Related Scores

Answer word length appears to have positive correlation with "answer_level_of_information", but does not seems to have any strong correlation on the plausibility or relevance of the the answers. Answer length can be a good indicator on the level of information of the answer, but does not provide much context on the correctness or helpfulness of the answers. Length of question_title is correlated with "question_body_critical" and length of question_body is anti-correlated with "question_well_written." Similarly, length seems not to be a good indicator to evaluate questions.

Figure 3 shows the distribution plots of five target variables. This tells us that most of the answers are rated very high on "answer_well_written", but rated low on "answer_type_procedure". The scores for "answer_type_instructions" and "answer_type_reason_explanation" seem to be equally distributed between samples.

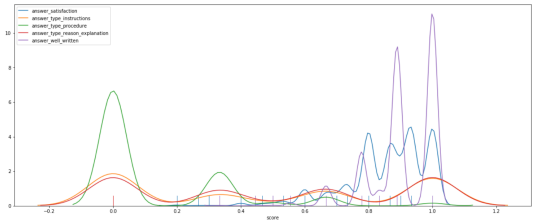

Figure 3: Examples of Answer Score Distribution

Figure 4 shows that the target scores can be different from one category to another. Most of the answers from culture sites are rated low on the the "answer_type_instruction", but answers from Stackoverflow and technology sites are rated higher on "answer_type_instruction". This makes sense because culture sites usually provide information instead of providing instructions. It also appears that "question_type_definition" and "question_type_consequence" are almost always 0 for all categories, indicating that some of the labels are quite imbalanced.

4 Methods

The purpose of the project is to build predictive algorithms for different subjective aspects of question-answering and give the answer a score base on its adequacy. I see this problem under two sub problems, first to embed/encode the text data, and second to use a model to find the correlations between predictor variables. I decided using Ridge Regression, as my baseline model, and BERT.

4.1 Models

The Ridge Regression is a variation of the Linear Regression model, which also called Tikhonov regularization. This is widely used as a method of regularization of ill-posed problems. This model is preferred to use in models with large numbers of parameters (multicollinearity). This means that the model can pick up correlation between word and word, character and character really quick and fast. Even though Ridge Regression model can do really well in picking up correlation between word and word, character and character, it can not pick up the correlation among phrases.

On the other hand, BERT model is a deeply bidirectional, unsupervised language representation, pre-trained using only a plain text corpus. In

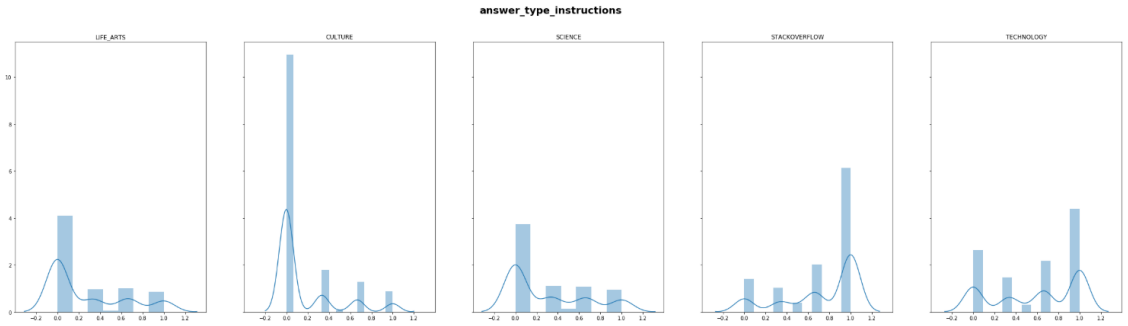

Figure 4: Answer_type_instructions Scores by Category

this case, Bert model convert all the word into vectors or matrix of vectors. Words with strong correlation with each other are usually represent as mostly the same vector. Therefore, in this case, where we need to pick up the phrase of word and determine if it is meaningful in different content, BERT seems to be a better model.

Both models are trained in the order of:

1. Data cleaning and pre-processing
2. Encoding/embedding text data into "number".
3. Fitting the model
4. Testing the results
5. Tuning the parameters

4.1.1 Baseline: Ridge Regression

Ridge model is basically an upgraded version of linear regression model with the flexible constant, alpha, to help reducing over fitting. In order for machine to understand the weighted correlation between words, I first needed to vectorize the text data. This was done by using TFIDF which gives a matrix of number representing the weighted term frequency as well as correlation of that word given the other words. This was then simply trained with Ridge Regression model.

For this model, I used all of the training samples to train the model. For cross-validation purposes, the training data was also splitted into two equal portions, then randomly splitted into smaller train and validation sets.

4.1.2 Bert Model:

Since BERT is a complex model, to reduce runtime, I started fitting model with only 0.05% of my data. In addition to lowering all the letters and removing stop words, as our data will not pass 32 bit, all the 64 bit data are transformed to 32 bit data to save the space as well as memory.

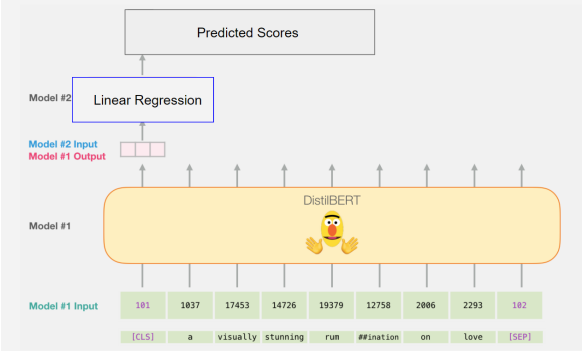

Figure 5: BERT Training Flow, Alammar[1]

The overall fitting process for this model can be found in Figure 5. The text data were first processed with 'bert-base-uncased', which has 12-layer, 768-hidden, 12-heads, 110M parameters and trained only on lower-cased English text. This step gave me an embedding table with vectors and matrices of 1 and 0 representing the correlation as well as the rating of each word in each content. For example, the word 'red' should have a strong correlation with 'blood' and the word 'blue' should have a strong correlation with 'sky'.

The next step was to get token for all the encoded data. These tokens replace the ids from the embedding table using BERT tokenizer. The data were then ready to fit with BERT model. In this process, BERT learns the correlation of the word to the other word in different contents. For example, if there is a question about a nocturnal animal, the model would pick the word mouse meaning rat than the computer mouse. This is the benefit of using BERT model that Ridge Regression does not have.

The output of BERT model then be used as an input for a linear regression model in order to output the predicted scores.

5 Results and Discussion

5.1 Ridge Regression Model:

There are 2 factors affecting the accuracy of this model: n-gram and alpha, the flexible value. In the tuning process, n-gram greater than 3 turned out to be over-fitting. The run-time was extremely long with higher value of n-gram. The model came out to give a best result with ngram = 3 and alpha = 2 (Figure 6).

	n_gram	Alpha	Accuracy_Rate
0	1	0	0.26
1	2	1	0.36
2	3	2	0.47
3	4	2	0.40
4	2	4	0.42

Figure 6: Ridge Regression Results

The benefits of using Ridge Regression are that it is very easy to train and gives us a good idea on how the data perform on a basic model. The run time is also relatively small compared to some other models. However, the accuracy rate remains low even with different parameter tuning. The main reason for this is that, in this problem, we not only need a meaningful sentence, but we also need a meaningful sentence in the right content. Ridge Regression does not have any layers to store the "definition" of a word in many different contexts. Ex: mouse(rat) vs. computer mouse. This model also does not understand the question if it is worded in a different way. Ex: it can understand: "how are you?" if this is in the data, but not "how is it going?" or "how are you doing?". In this case, Ridge Regression fails to give an accurate prediction.

5.2 Bert Model

Using only 0.05% of my data for initial testing, with groupKFold = 5, I got a result of 0.42 loss of epoch 3/3 on average. The amount of training data was then gradually increased from 0.05% to 1%. The reason for not using a large amount of data for this model is that BERT is a tremendous model and the run-time is also extremely long. Also, the main purpose is not to get a best result, but to see how the model performs and if the loss comes down by simply increase the amount of data and the number of layers.

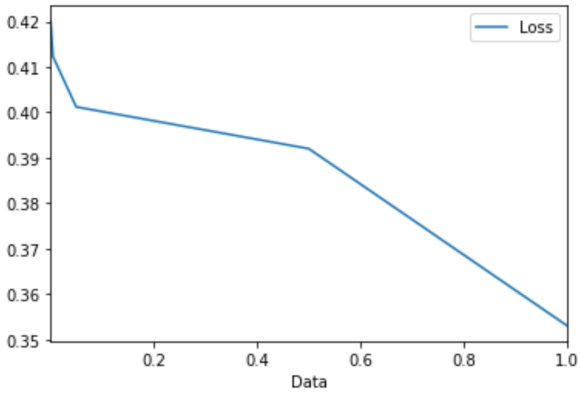

Figure 7: BERT Loss by % of Data Used

It turns out that the loss is decreased with larger amount of training data. The loss also dropped significantly with higher number of layers. As each layer represents the definition and the meaning of the word in each content, with more layer, the model would be able to do a better job picking the correct word for correct content. Overall, BERT model seems to perform very well. With only 1% of the data, I was able to get the loss down to to around 0.355. BERT turns out to be a very promising model for developing a evaluating metrics for Question Answering systems.

References

[1] Alammar, Jay. 2019. *A Visual Guide to Using BERT for the First Time.* Visualizing Machine Learning One Concept at a Time.

[2] Chen, Anthony, et al 2019. *Evaluating Question Answering Evaluation.* mrqa.github.io/assets/papers/45_Paper.pdf.

[3] Clarke, C.L. 2009. *Web question answering*. Encyclopedia of DatabaseSystems, Springer, New York, NY.

[4] Lopez, V., Uren, V., Sabou, M. and Motta, E. 2011. *Is question answering fit for the semantic web?: a survey*. Semantic Web, Vol. 2 No. 2

[5] Strzalkowski, T. and Harabagiu, S. (Eds) 2006. *Advances in Open Domain Question Answering,Vol. 32*. Springer Science Business Media, Dordrecht.

[6] Zhao, Yiming and Zhang, Jin and Xia, Xue and Le, Taowen 2018. *Evaluation of Google question-answering quality*. Library Hi Tech.

[7] Zhu, Z., Bernhard, D., Gurevych, I. 2009. *Multi-dimensional Model for Assessing the Quality of Answers in Social QA Sites.* Technical Report TUD-CS-2009-0158. Technische Universität Darmstad.