BANA 8083-005 Project Report

Google QUEST Q&A Labeling

Prepared by Jagruti Joshi (M13840197)

Committee Members: Dr. Yan Yu, <second reader>

Table of Contents

[Abstract 3](#_Toc44876073)

[Introduction 4](#_Toc44876074)

[Dataset 5](#_Toc44876075)

[Exploratory Data Analysis 5](#_Toc44876076)

[The BERT Model 11](#_Toc44876077)

[Introduction 11](#_Toc44876078)

[History 12](#_Toc44876079)

[Applications 12](#_Toc44876080)

[Architecture 13](#_Toc44876081)

[Data Preprocessing 14](#_Toc44876082)

[Model Pre-training 15](#_Toc44876083)

[Model Fine-tuning 16](#_Toc44876084)

[Model Hyperparameter Tuning 17](#_Toc44876085)

[Model Results 17](#_Toc44876086)

[The DistilBERT Model 18](#_Toc44876087)

[Introduction 18](#_Toc44876088)

[Architecture 18](#_Toc44876089)

[Model Hyperparameter Tuning 19](#_Toc44876090)

[Model Results 20](#_Toc44876091)

[BERT Model vs DistilBERT Model 20](#_Toc44876092)

[Conclusion and Next Steps 21](#_Toc44876093)

[References 22](#_Toc44876094)

## Abstract

Computers are good at answering questions with single, verifiable answers. But humans are often still better at answering questions about opinions, recommendations, or personal experiences. Humans are better at addressing subjective questions that require a deeper, multidimensional understanding of context - something computers are not trained to do well yet. Questions can take many forms - some have multi-sentence elaborations while 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.

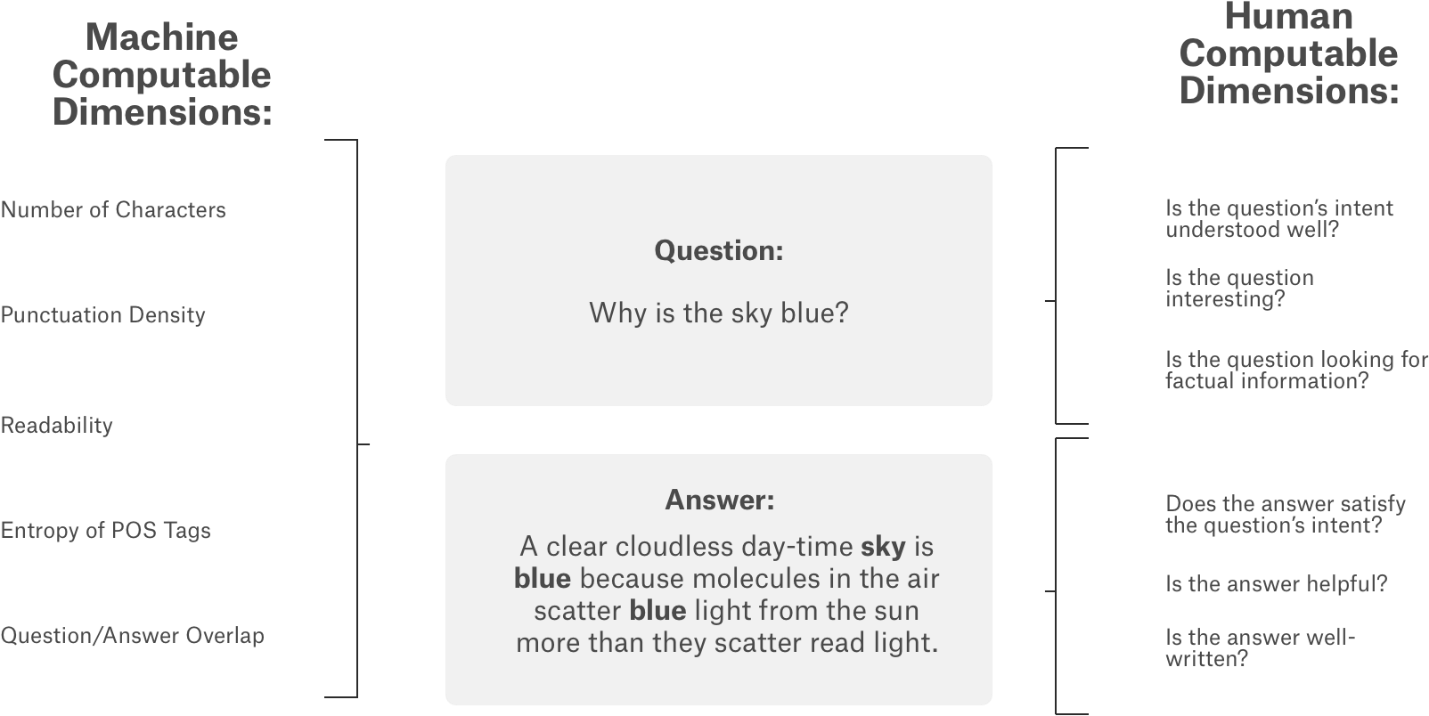
Unfortunately, it is hard to build better subjective question-answering algorithms because of a lack of data and predictive models. That is why the CrowdSource team at Google Research, a group dedicated to advancing NLP and other types of ML science via crowdsourcing, has collected data on a number of these quality scoring aspects.

This project aims to use the new dataset to build predictive algorithms for different subjective aspects of question-answering and improve automated understanding of complex question answer content. We will be focusing on Bidirectional Encoder Representations from Transformers (BERT) and DistilBERT, a distilled version of BERT that is smaller, faster, cheaper, and lighter. [1]

## Introduction

Question answering (QA) is a classical problem in Natural Language Processing (NLP). QA problems are concerned with building systems that use data to automatically answer questions posed by humans in a natural language. These QA systems find applications in retrieving useful information for a large-scale text corpus which are still difficult and time-consuming for humans. [2] And with advancement in computer science and machine learning, computers have gotten so good at QA problem that they are outperforming humans. However, subjective QA problems, a type of QA that involves answering questions about opinions, recommendations, or personal experiences is an area where humans still outperform computers.

*Figure 1: Computer vs Humans*



Subjective QA problems were difficult for computers mainly due to the lack of data and predictive models. The CrowdSource team at Google Research, a group dedicated to advancing NLP and other types of ML science via crowdsourcing, has solved the data side of this problem by collecting data on a number of quality scoring aspects related to both questions as well as answers. They have worked on developing the Google Quest Q&A Labeling problem for nearly 2 years. In January 2020, the team posted the data on Kaggle as part of the Google Quest Challenge.

The team collected a wide variety of questions from various StackExchange properties and picked some of the answers to those questions (the picked answers were not necessarily the most upvoted ones). The data consisted of questions across categories such as culture, life arts, science, technology and stackoverflow (computer programming). Within these broad categories, the team collected data for different types of questions such as comparison, choice, consequence, definition, entity, instructions, procedure, reason explanation and spelling questions.

The data consists of over 1 million target label values. Demonstrating these subjective labels can be predicted reliably can shine a new light on this research area. Results from the Google Quest Q&A Labeling Kaggle competition will inform the way future intelligent Q&A systems will get built, hopefully contributing to them becoming more human-like.

## Dataset

The data for this project includes questions and answers from various StackExchange properties. The task is to predict target values of 30 labels for each question-answer pair. The data can be found here: <https://www.kaggle.com/c/google-quest-challenge/data>.

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. By lessening our dependency on complicated and opaque rating guidelines, the team is hoping to increase the re-use value of this data set.

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. [1]

## Exploratory Data Analysis

Preliminary data analysis showed that 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.

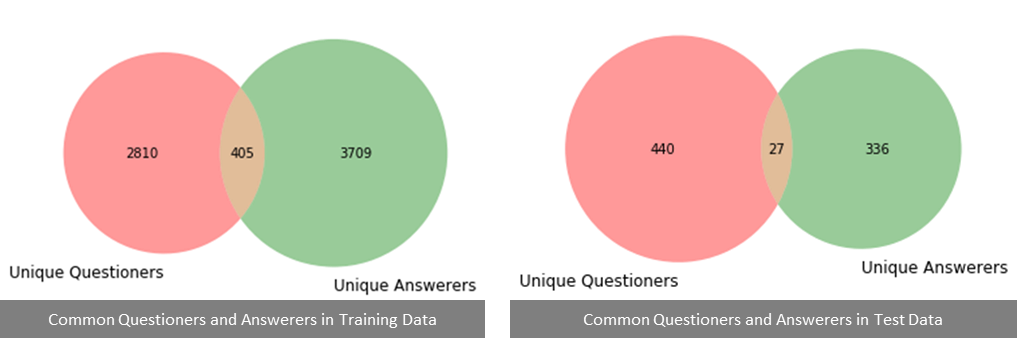
There are no missing values in the training data and the test data. Training data has 21 target labels related to questions and 9 target labels related to answers. Target labels are not present in the test data. Further results from the preliminary data analysis are summarized in the table below:

*Table 1: Preliminary Data Analysis Summary*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Dataset** | **Shape** | **Unique Categories** | **Unique Hosts** | **Unique Questioners** | **Unique Answerers** |
| Training | 6079 x 41 | 5 | 63 | 3215 | 4114 |
| Test | 476 x 11 | 5 | 55 | 467 | 363 |

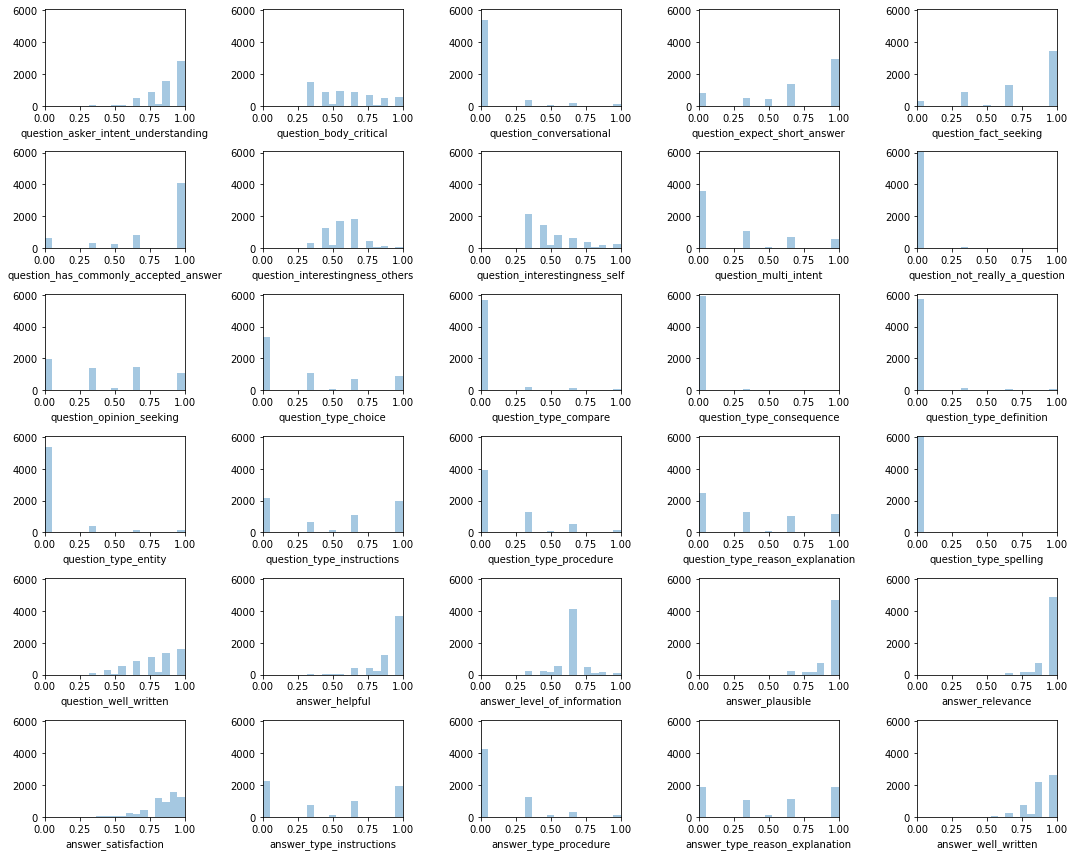
Overlap is observed between unique questioners and unique answerers in the training data and the test data i.e. few users ask questions as well as answer questions asked by other users.

*Figure 2: Common Questioners and Answerers in Training and Test Data*



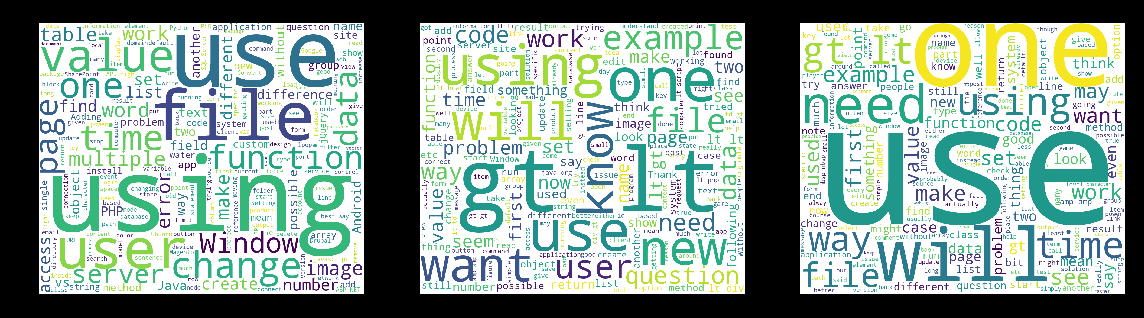
The distribution of each of the 30 target labels is shown in Figure 2. Compare, consequence, definition, entity, and spelling questions are hardly present in the training data.

*Figure 3: Distribution of Target Labels in Training Data*

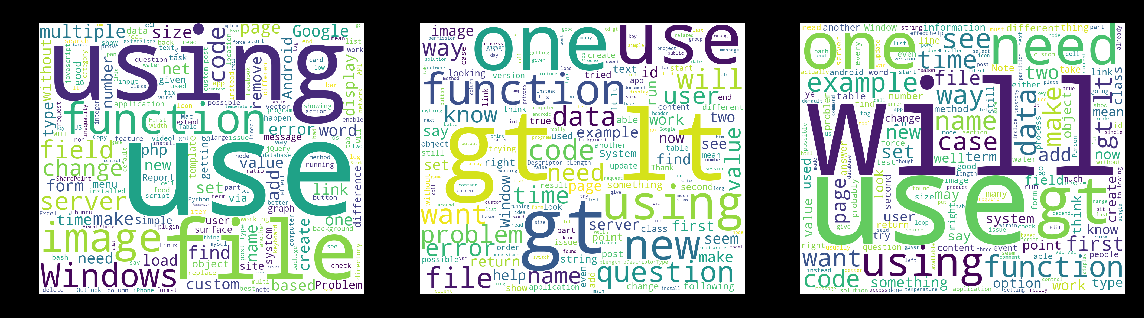
5

Word clouds for question title, question body and answer fields in the training data and the test data indicate most frequently used words by questioners and answerers.

*Figure 4: Training Data Word Cloud*

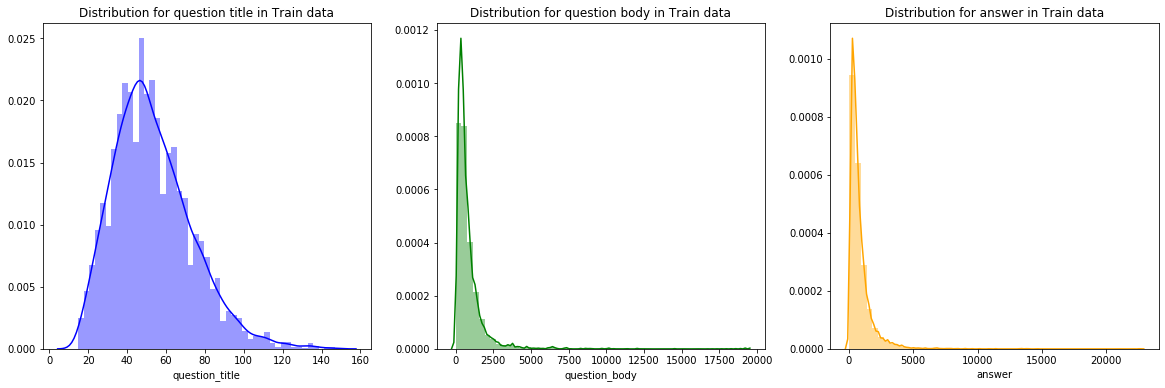


*Figure 5: Test Data Word Cloud*



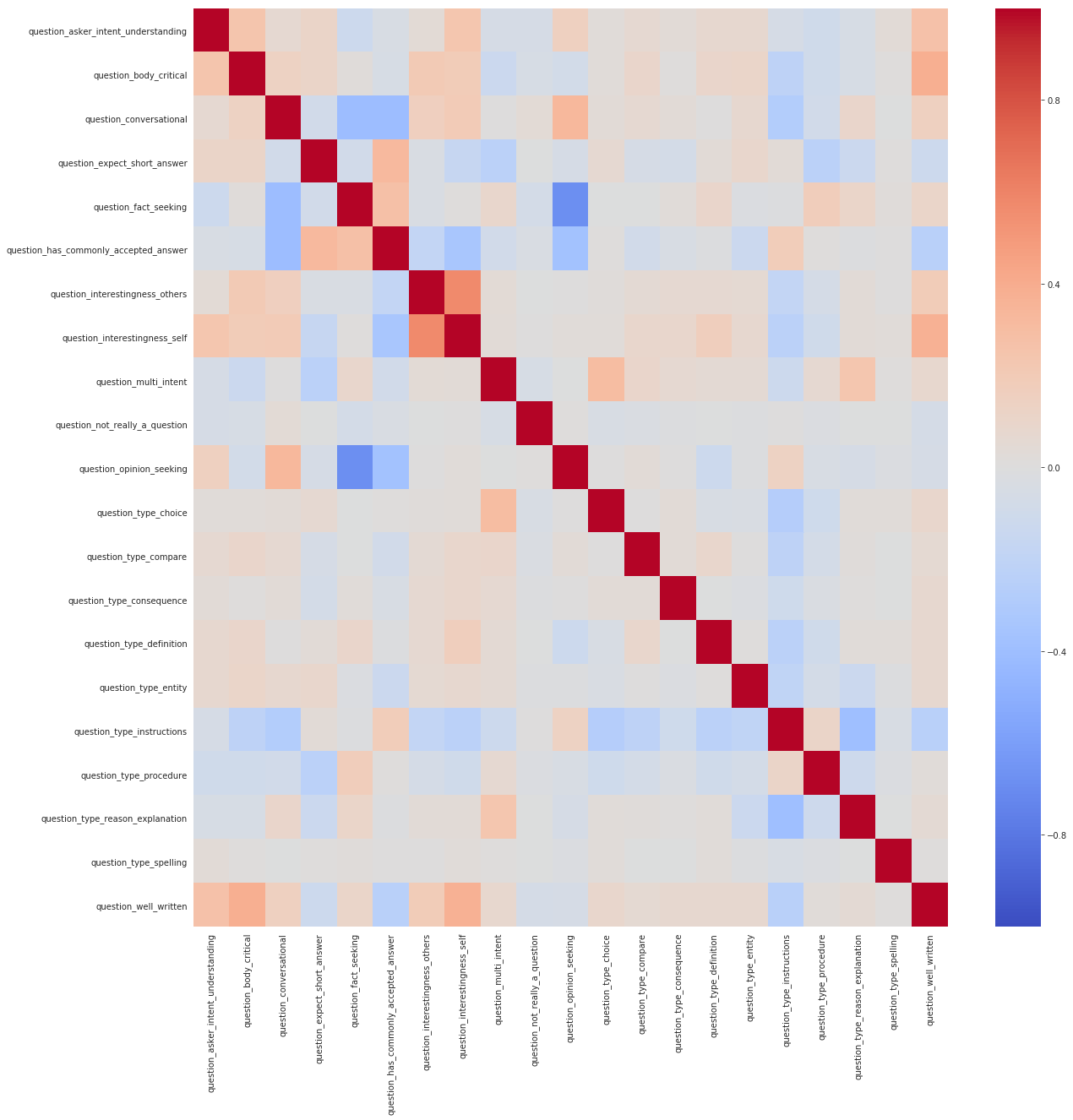
Majority of the questions have <2500 words while majority of the answers have <5000 words.

*Figure 6: Distribution of length of questions and answers in training data*



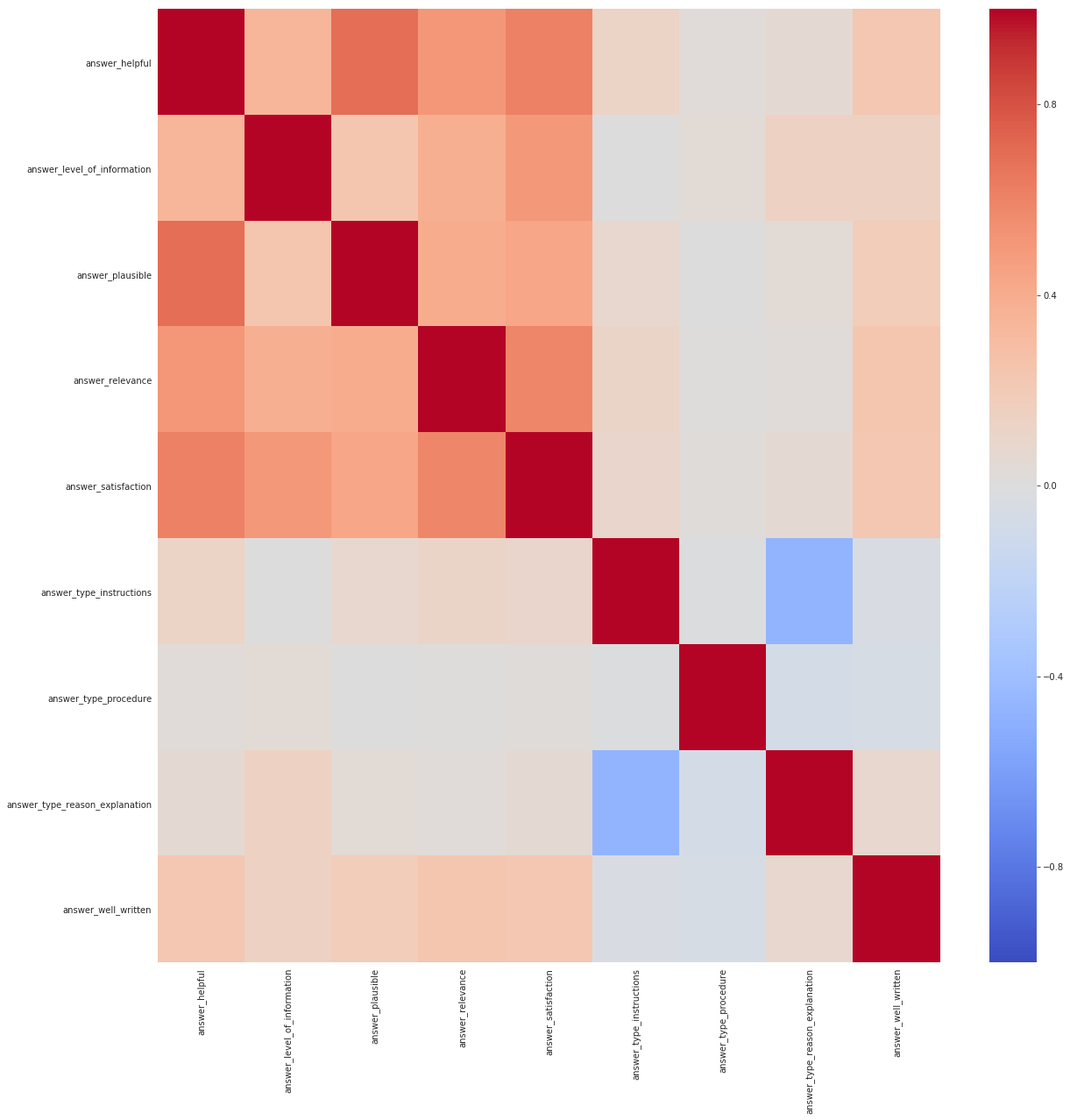
Target labels question interestingness self and others are highly correlated. High correlation is also observed between interesting questions and well-written questions.

*Figure 7: Question Target Labels Correlation Plot*



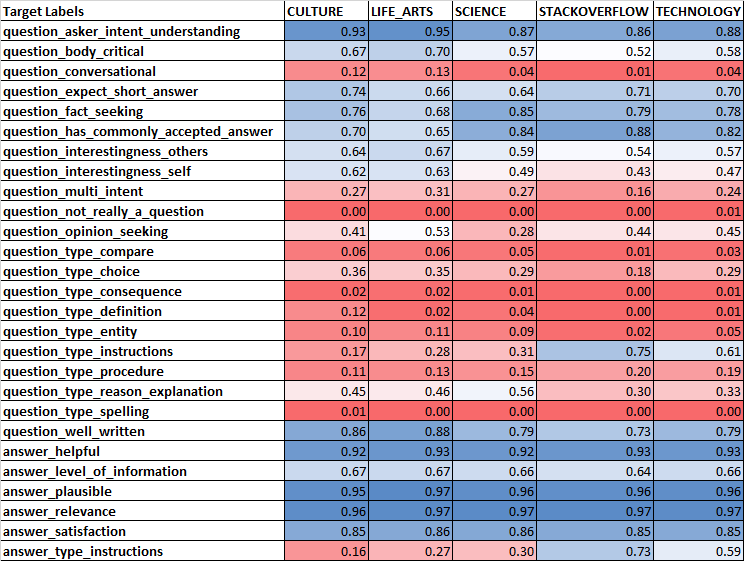
High correlation is observed between plausible answers, helpful answers, and satisfactory answers.

*Figure 8: Answer Target Labels Correlation Matrix*



Raters have rated questions to be more conversational and interesting in culture and life arts categories. Instruction type questions and answers occur on Stackoverflow more than in any other category.

*Table 2: Average Target Labels by Category*



## The BERT Model

### Introduction

BERT stands for Bidirectional Encoder Representation from Transformers. It is a state-of-the-art deep learning model developed by Google for NLP. It was created and published by Jacob Devlin and his colleagues from Google. The original English-language BERT model used two corpora in pre-training: BookCorpus, a popular large-scale text corpus and the English Wikipedia which currently host over 6 million articles. This pre-training along with the ability to easily fine-tune BERT for specific tasks has made it a widely used model in the NLP community. [3]

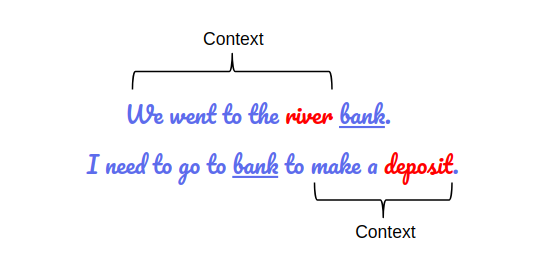
When BERT was published, it achieved state-of-the-art performance on several NLP tasks including the SQuAD (Stanford Question Answering Dataset) v1.1 and v2.0 which is one of the primary reasons why it was chosen for this project. Although both these problems fall under the same category of Question answering (QA) in the field of NLP, the end goals are different. SQuAD aims to find the answer to a question given the question and a paragraph containing the answer whereas Google Quest Q&A Labeling deals with predicting subjective target labels associated with both questions and answer given a question-answer pair.

### History

The idea of using pre-trained models trained on a large-scale text corpus was not new. Some examples of such models are Semi-supervised Sequence Learning, Generative Pre-Training, ELMo, and ULMFit. The key differences between the previous models and BERT was that it is unsupervised and deeply bidirectional. BERT learns information from both the left as well as the right side of a token’s context during the training phase.

Context-free models such as word2vec or GloVe generate a single word embedding representation for each word in the vocabulary, where BERT considers the context for each occurrence of a given word. For instance, whereas the vector for "running" will have the same word2vec vector representation for both of its occurrences in the sentences "He is running a company" and "He is running a marathon", BERT will provide a contextualized embedding that will be different according to the sentence. [3]

*Figure 9: Contextualized Embedding in BERT*



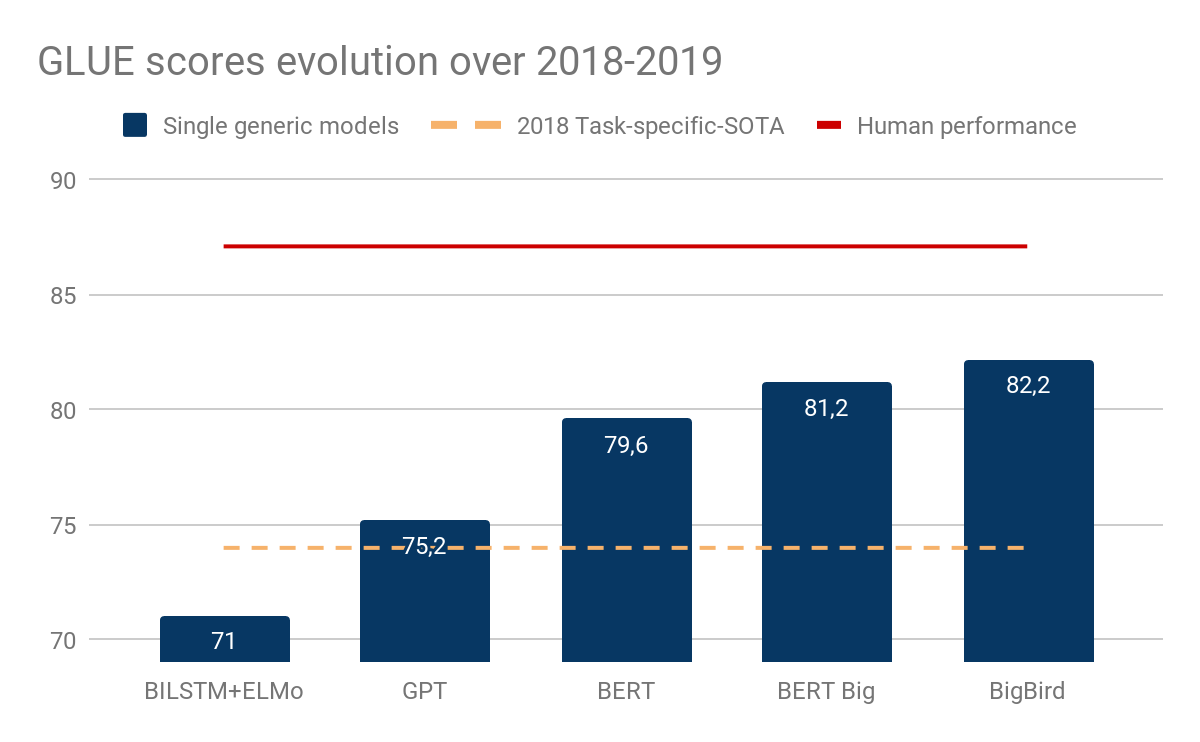
### Applications

BERT has significantly altered the NLP landscape by achieving state-of-the-art results on 11 individual NLP tasks with just a little fine-tuning. It has inspired many recent NLP architectures, training approaches and language models, such as Google’s TransformerXL, OpenAI’s GPT-2, XLNet, ERNIE2.0, RoBERTa, etc. [4]

Google is leveraging BERT to better understand user searches. On October 25, 2019, Google Search announced that they had started applying BERT models for English language search queries within the US. On December 9, 2019, it was reported that BERT had been adopted by Google Search for over 70 languages. [3] Google believes this step (or progress in natural language understanding as applied in search) represents “the biggest leap forward in the past five years, and one of the biggest leaps forward in the history of Search”.

The following graph shows the evolution of scores for GLUE benchmark — the average of scores in various NLP evaluation tasks.

*Figure 10: GLUE Scores Evolution with NLP Advancements*



While it’s not clear that all GLUE tasks are very meaningful, generic models based on an encoder named Transformer (Open-GPT, BERT and BigBird), closed the gap between task-dedicated models and human performance and within less than a year. [4]

### Architecture

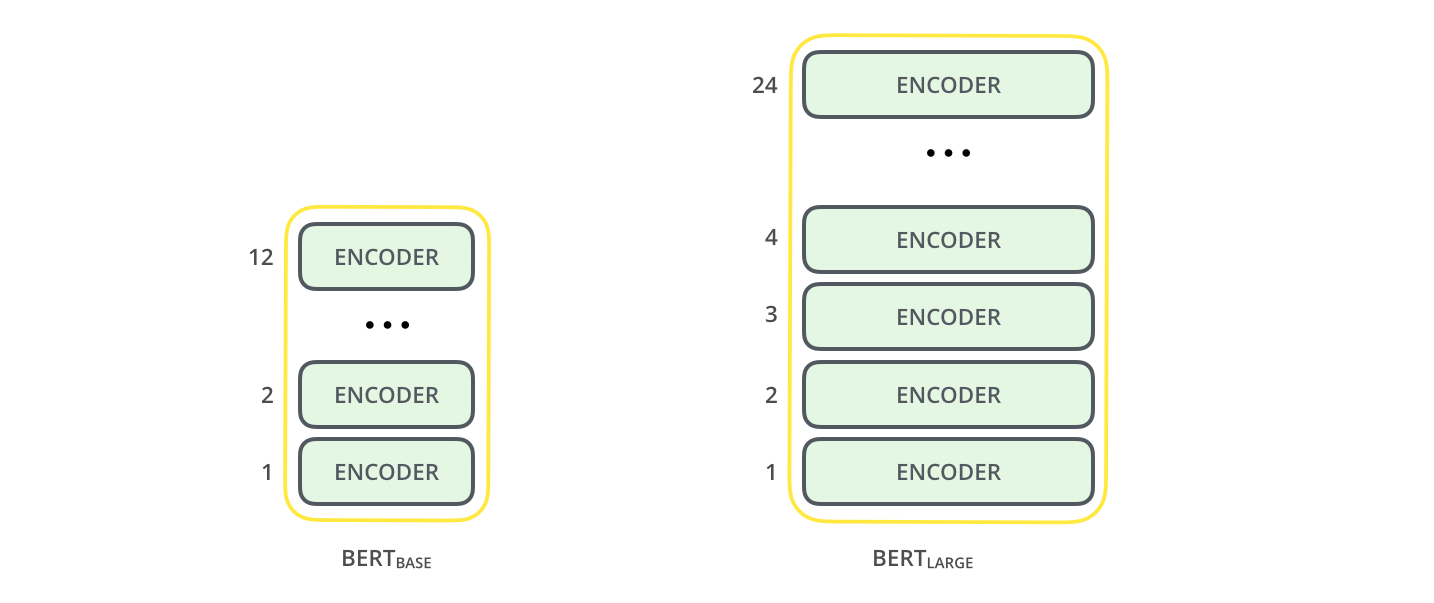
BERT relies on a Transformer (the attention mechanism that learns contextual relationships between words in a text). A basic Transformer consists of an encoder to read the text input and a decoder to produce a prediction for the task. Since BERT’s goal is to generate a language representation model, it only needs the encoder part. It is basically a bunch of Transformer encoders stacked together. The concept of bidirectionality is the key differentiator between BERT and its predecessor, OpenAI GPT. BERT is bidirectional because its self-attention layer performs self-attention on both directions.

There are two types of pre-trained versions of BERT depending on the scale of the model architecture – BERT-Base and BERT-Large. The key differences between these architectures is explained in the table below: [5]

*Table 3: BERT-Base vs BERT-Large*

|  |  |
| --- | --- |
| **BERT-Base** | **BERT-Large** |
| Comparable in size to the OpenAI Transformer in order to compare performance | A ridiculously huge model which achieved the state-of-the-art results reported in the Google paper |
| pre-trained on 4 cloud TPUs for 4 days | pre-trained on 16 TPUs for 4 days |
| 12-layer | 24-layer |
| 768-hidden-nodes | 1024-hidden-nodes |
| 12-attention-heads | 16-attention-heads |
| 110M parameters | 340M parameters |

*Figure 11: BERT Encoders*



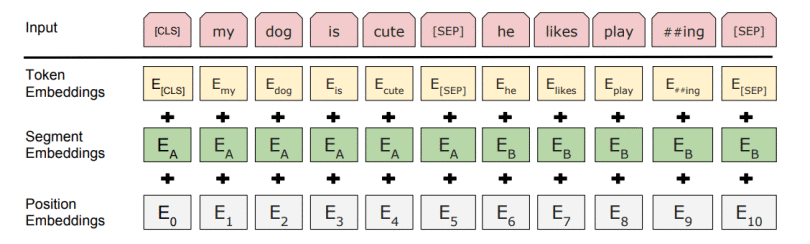
Data Preprocessing

The input representation used by BERT is able to represent a single text sentence as well as a pair of sentences (eg., Question, Answering) in a single sequence of tokens. These tokens are first converted into vectors and then processed in the neural network. But before processing can start, BERT needs the input to be massaged and decorated with some extra metadata:

1. **Token embeddings**: A [CLS] token is added to the input word tokens at the beginning of the first sentence and a [SEP] token is inserted at the end of each sentence.
2. **Segment embeddings**: A marker indicating Sentence A or Sentence B is added to each token. This allows the encoder to distinguish between sentences.
3. **Positional embeddings**: A positional embedding is added to each token to indicate its position in the sentence.

The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

*Figure 12: The input representation for BERT*



Essentially, the Transformer stacks a layer that maps sequences to sequences, so the output is also a sequence of vectors with a 1:1 correspondence between input and output tokens at the same index. [5]

### Model Pre-training

Pre-training makes use of the following two strategies:

1. **Masked LM (MLM)**

The idea here is “simple”: Randomly mask out 15% of the words in the input — replacing them with a [MASK] token — run the entire sequence through the BERT attention based encoder and then predict only the masked words, based on the context provided by the other non-masked words in the sequence. However, there is a problem with this naive masking approach — the model only tries to predict when the [MASK] token is present in the input, while we want the model to try to predict the correct tokens regardless of what token is present in the input. To deal with this issue, out of the 15% of the tokens selected for masking:

* 80% of the tokens are actually replaced with the token [MASK].
* 10% of the time tokens are replaced with a random token.
* 10% of the time tokens are left unchanged.

While training the BERT loss function considers only the prediction of the masked tokens and ignores the prediction of the non-masked ones. This results in a model that converges much more slowly than left-to-right or right-to-left models.

1. **Next Sentence Prediction (NSP)**

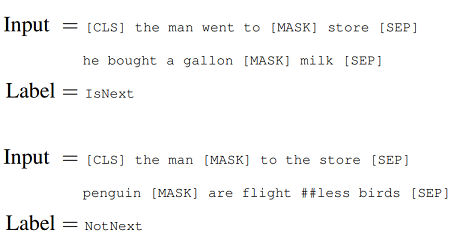
In order to understand relationship between two sentences, BERT training process also uses next sentence prediction. A pre-trained model with this kind of understanding is relevant for tasks like question answering. During training the model gets as input pairs of sentences and it learns to predict if the second sentence is the next sentence in the original text as well.

As we have seen earlier, BERT separates sentences with a special [SEP] token. During training the model is fed with two input sentences at a time such that:

* 50% of the time the second sentence comes after the first one
* 50% of the time it is a a random sentence from the full corpus

BERT is then required to predict whether the second sentence is random or not, with the assumption that the random sentence will be disconnected from the first sentence:

*Figure 13: Second Sentence Prediction using BERT*



To predict if the second sentence is connected to the first one or not, basically the complete input sequence goes through the Transformer based model, the output of the [CLS] token is transformed into a 2×1 shaped vector using a simple classification layer, and the IsNext-Label is assigned using softmax.

The model is trained with both Masked LM and Next Sentence Prediction together. This is to minimize the combined loss function of the two strategies — “together is better”.

[5]

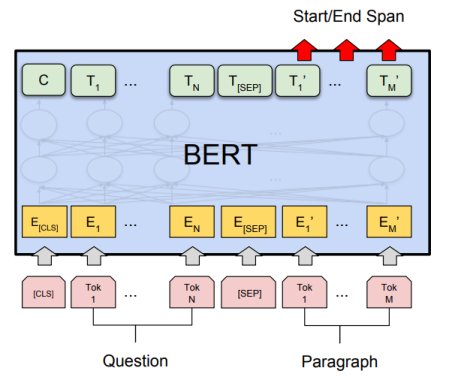
### Model Fine-tuning

Using BERT for a specific task is relatively straightforward. BERT can be used for a wide variety of language tasks, while only adding a small layer to the core model.

For QA tasks, given a question and a context paragraph, the model predicts a start and an end token from the paragraph that most likely answers the question.

Just like sentence pair tasks, the question becomes the first sentence and paragraph the second sentence in the input sequence. There are only two new parameters learned during fine-tuning a start vector and an end vector with size equal to the hidden shape size. The probability of token i being the start of the answer span is computed as – softmax(S . K), where S is the start vector and K is the final transformer output of token i. The same applies to the end token. [4]

*Figure 14: BERT Fine-tuning for QA task*



### Model Hyperparameter Tuning

The optimal hyperparameter values are task specific. But the authors of the BERT paper found that the following range of values works well across all tasks:

1. Dropout – 0.1
2. Batch Size – 16, 32
3. Learning Rate (Adam) – 5e-5, 3e-5, 2e-5
4. Number of epochs – 3, 4

The authors also observed that large datasets (> 100k labeled samples) are less sensitive to hyperparameter choice than smaller datasets. [4]

### Model Results

Question title, question body and answer are used as features for the BERT model. Cross-validation technique is used divide the training data into 5 folds. Each fold is trained for 4 epochs (To ensure the code runs within 2 hours, cross-validation is done only thrice, and the average of the results is reported).

Models are 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 reported for each fold and epoch in the figure below:

*Figure 15: Overall Spearman's ρ from BERT Model*

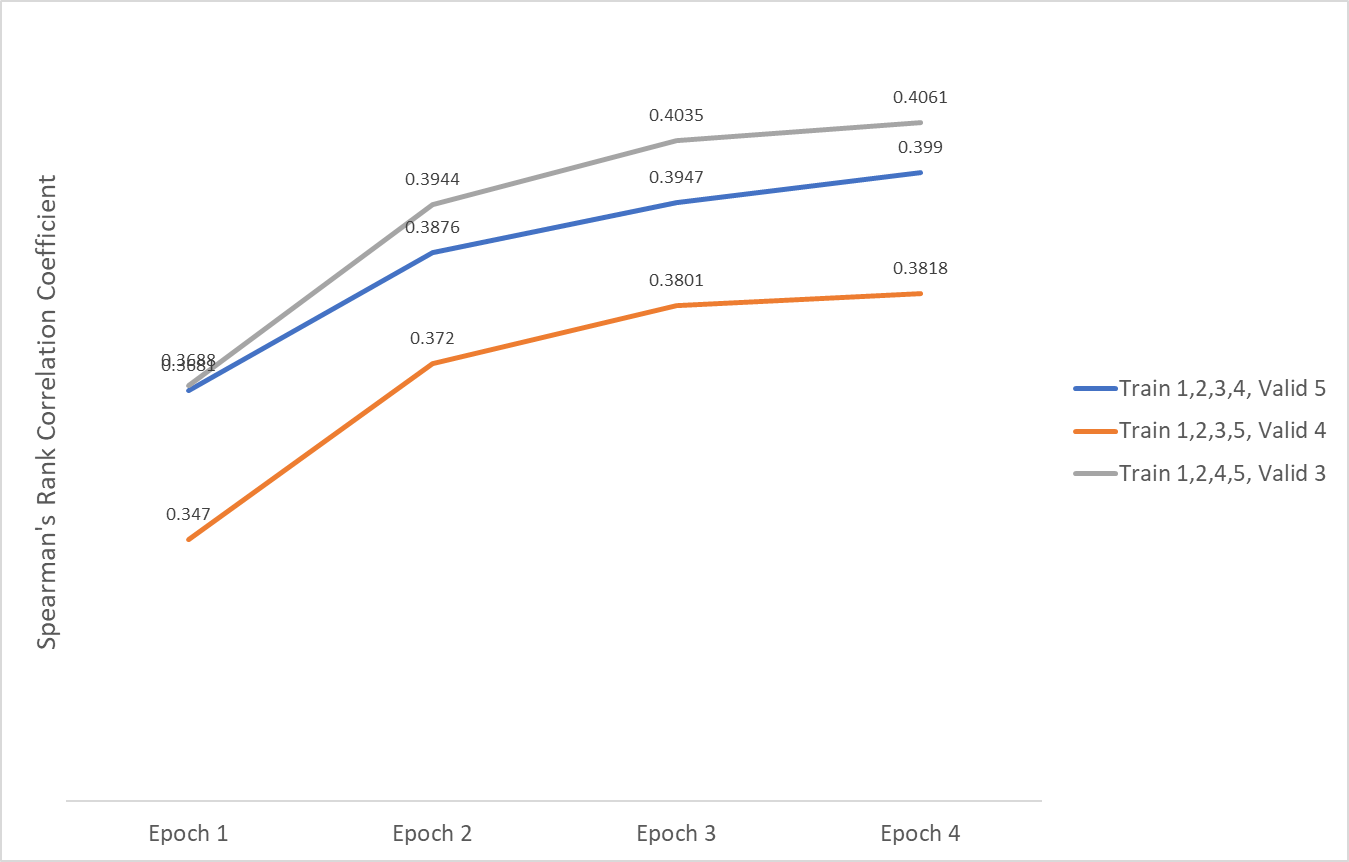
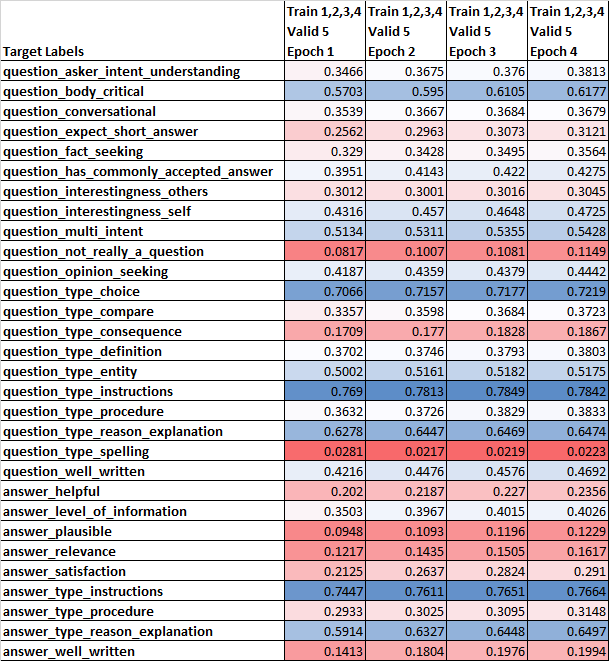


Table 4: *Spearman's ρ by Target Label for Fold 1 from BERT Model*



The BERT model predicts 8 out of the 30 target labels well i.e. Spearman's correlation coefficient > 0.5. For instruction and reason explanation type questions and answers, Spearman's correlation coefficient > 0.6

## The DistilBERT Model

### Introduction

DistilBERT is a distilled version of the BERT model that is not only smaller but also faster, cheaper, and lighter. It was developed and open sourced by the team at HuggingFace. DistilBERT like its larger counterpart can be easily fine-tuned to achieve good performance on a wide variety of NLP tasks including QA. Knowledge distillation is leveraged during the pre-training phase to reduce the size of a BERT model by 40%, while retaining 97% of its language understanding capabilities. This new DistilBERT model is 60% faster. To leverage the inductive biases learned by larger models during pre-training, a triple loss combining language modeling, distillation and cosine-distance losses is introduced. [14]

DistilBERT’s history and applications are pretty similar to that of BERT.

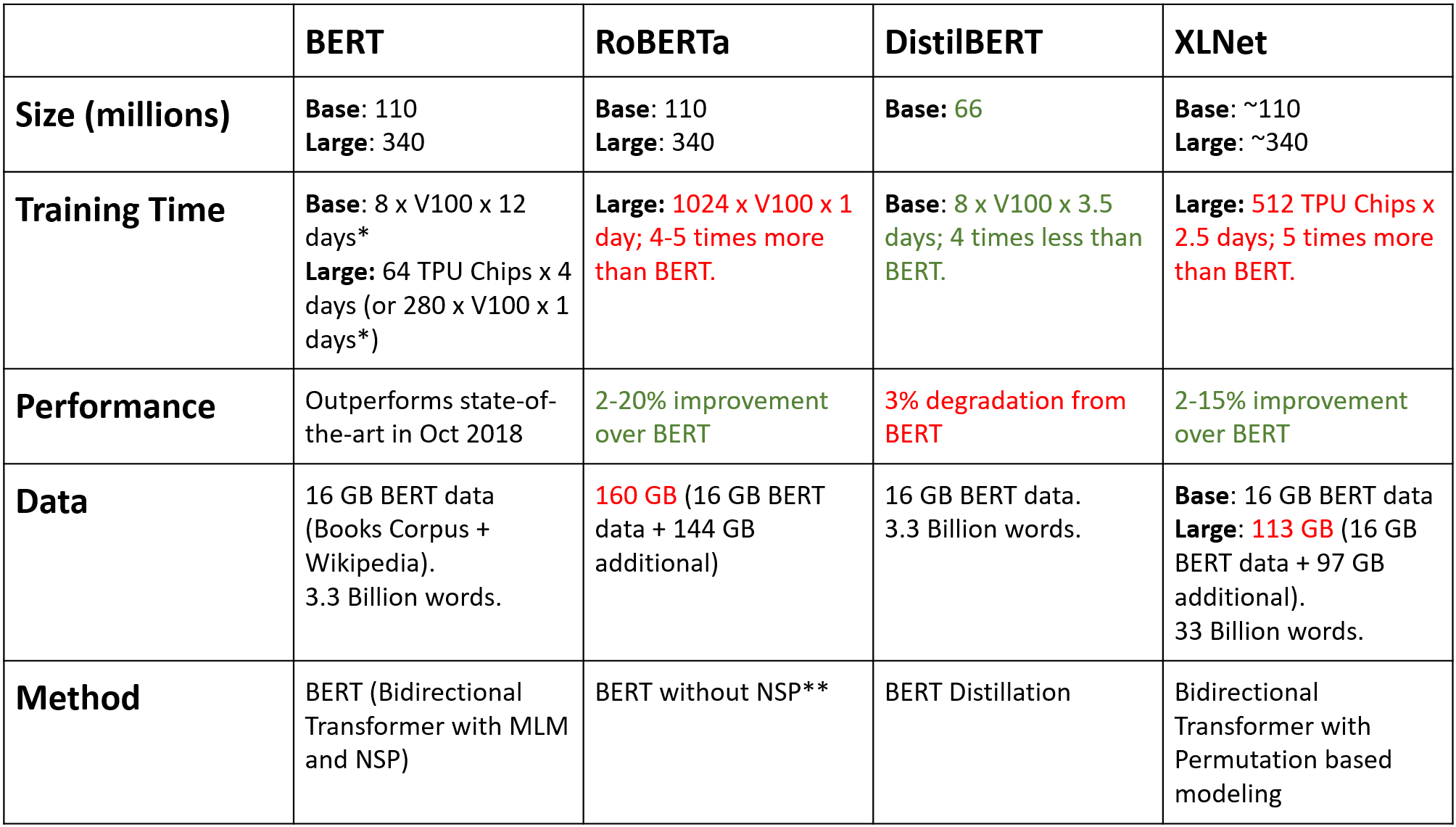
### Architecture

DistilBERT uses a technique called distillation, which approximates the Google’s BERT, i.e. the large neural network by a smaller one. The idea is that once a large neural network has been trained, its full output distributions can be approximated using a smaller network. This is in some sense similar to posterior approximation. One of the key optimization functions used for posterior approximation in Bayesian Statistics is Kulback Leiber divergence and has naturally been used here as well. [15]

Knowledge distillation [Bucila et al., 2006, Hinton et al., 2015] is a compression technique in which a compact model - the student - is trained to reproduce the behaviour of a larger model - the teacher - or an ensemble of models.

The student - DistilBERT - has the same general architecture as BERT. The token-type embeddings and the pooler are removed while the number of layers is reduced by a factor of 2. Most of the operations used in the Transformer architecture (linear layer and layer normalisation) are highly optimized in modern linear algebra frameworks and our investigations showed that variations on the last dimension of the tensor (hidden size dimension) have a smaller impact on computation efficiency (for a fixed parameters budget) than variations on other factors like the number of layers. Thus we focus on reducing the number of layers.

*Table 5: DistilBERT vs BERT vs Other Models*



DistilBERT’s data preprocessing, model pre-training and model fine-tuning steps are pretty similar to that of BERT.

### Model Hyperparameter Tuning

The optimal hyperparameter values are task specific. But the authors of the DistilBERT paper found that the following range of values works well across all tasks:

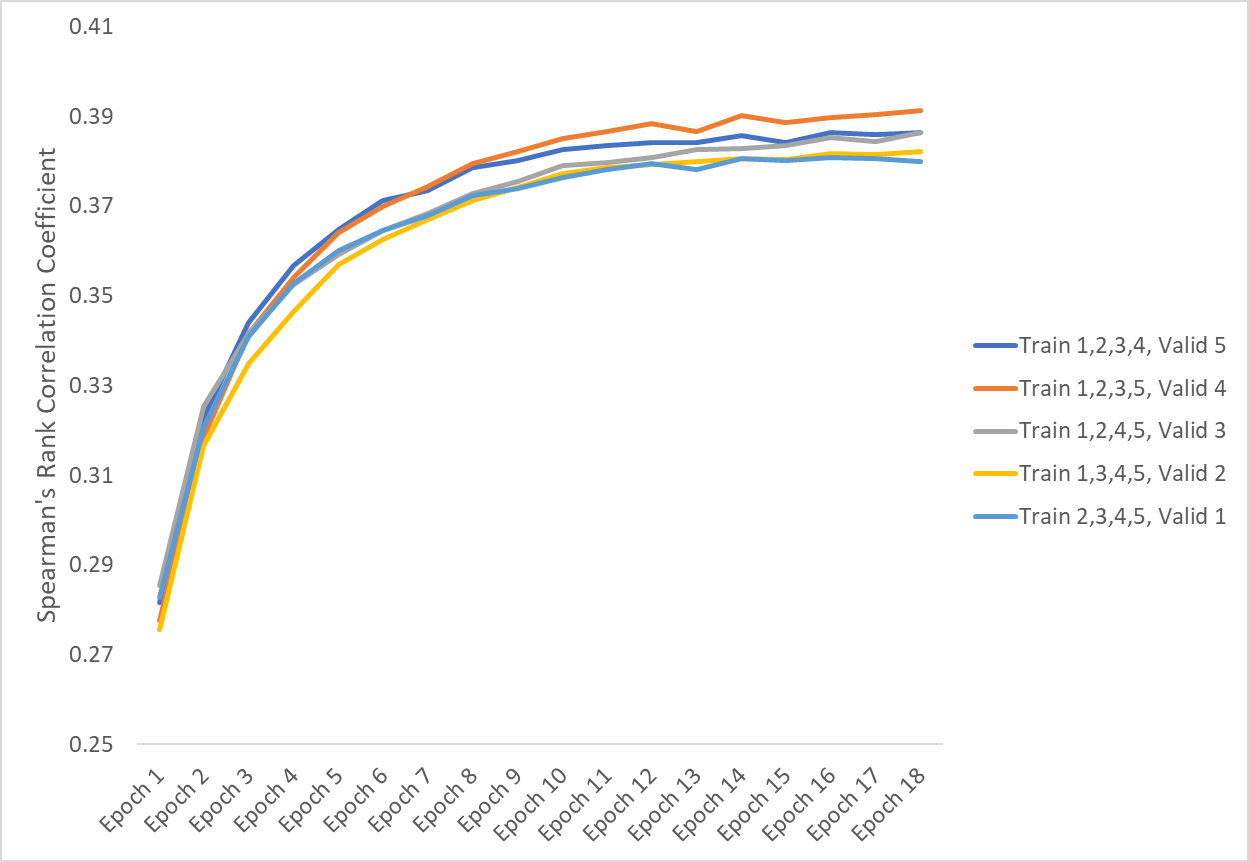
1. Batch Size – 16, 32 (more than BERT)
2. Learning Rate (Adam) – 1e-4 (faster than BERT)
3. Number of epochs – 18, 19, 20 (more than BERT)

### Model Results

Question title, question body and answer are used as features for the BERT model. Cross-validation technique is used divide the training data into 5 folds. Each fold is trained for 18 epochs.

Models are 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 reported for each fold and epoch in the figure below:

*Figure 15: Spearman's ρ from DistilBERT Model*



## BERT Model vs DistilBERT Model

*Table 6: Model Training – BERT vs DistilBERT*

|  |  |  |
| --- | --- | --- |
| **Parameter** | **BERT** | **DistilBERT** |
| Max. Sequence Length | 384 | 512 |
| No. of Folds | 5 | 5 |
| No. of Folds Utilized | 3 | 5 |
| Training Samples | 4863 | 4863 |
| Validation Samples | 1216 | 1216 |
| No. of Epochs | 4 | 18 |
| Adam’s Learning Rate | 3e-5 | 1e-4 |
| Batch Size | 6 | 32 |
| Loss Function | binary crossentropy | binary crossentropy |
| Model Runtime | ~2 hrs | ~2 hrs |

*Table 6: Model Evaluation – BERT vs DistilBERT*

|  |  |  |
| --- | --- | --- |
| **Parameter** | **BERT** | **DistilBERT** |
| Average Spearman's ρ over all folds | 0.3933 | 0.384 |

Since DistilBERT’s performance is almost equal to BERT’s performance in the 2-hour runtime limit, DistilBERT is selected as the final model for this problem.

## Conclusion and Next Steps

BERT and DistilBERT are undoubtedly a breakthrough in the use of Machine Learning for Natural Language Processing. The fact that these models are approachable and allows fast fine-tuning will likely allow a wide range of practical applications in the future. Other transformed based models such as RoBERTa and XLNet can also be applied to the Google Quest Q&A Labeling problem.

## References

[1] <https://www.kaggle.com/c/google-quest-challenge> - Google QUEST Q&A Labeling competition page on Kaggle

[2] <https://en.wikipedia.org/wiki/Question_answering> - Question answering article on Wikipedia

[3] <https://en.wikipedia.org/wiki/BERT_(language_model)> - BERT (language model) article on Wikipedia

[4] <https://www.kaggle.com/abhinand05/bert-for-humans-tutorial-baseline-version-2> - BERT for Humans: Tutorial+Baseline (Version 2) notebook on Kaggle

[5] <https://towardsml.com/2019/09/17/bert-explained-a-complete-guide-with-theory-and-tutorial/> - BERT Explained: A Complete Guide with Theory and Tutorial on Towards Machine Learning blog

[6] <https://arxiv.org/pdf/1910.01108.pdf> - DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter

[7] <https://towardsdatascience.com/bert-roberta-distilbert-xlnet-which-one-to-use-3d5ab82ba5f8> - BERT, RoBERTa, DistilBERT, XLNet — which one to use?

[8] <https://www.kaggle.com/abhishek/distilbert-use-features-oof> - DistilBERT + USE + Features + OOF notebook on Kaggle

[9] <http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/> - A Visual Guide to Using BERT for the First Time on Jay Alammar’s blog

[10] <http://jalammar.github.io/illustrated-bert/> - The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning) on Jay Alammar’s blog

[11] <https://towardsdatascience.com/bert-roberta-distilbert-xlnet-which-one-to-use-3d5ab82ba5f8> - BERT, RoBERTa, DistilBERT, XLNet — which one to use? on Towards Data Science blog

[12] <https://medium.com/huggingface/distilbert-8cf3380435b5> - Smaller, faster, cheaper, lighter: Introducing DistilBERT, a distilled version of BERT on Medium – Victor Sanh’s blog

[13] <https://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient> - Spearman's rank correlation coefficient article on Wikipedia

[14] <https://www.youtube.com/watch?v=iDulhoQ2pro> - Attention Is All You Need video by Yannic Kilcher

[15] <https://www.youtube.com/watch?v=-9evrZnBorM> - BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding video by Yannic Kilcher

[15] <https://www.youtube.com/playlist?list=PLam9sigHPGwOBuH4_4fr-XvDbe5uneaf6> – BERT Research Series video playlist by ChrisMcCormickAI

[16] <https://www.udemy.com/course/bert-nlp-algorithm/> - Learn BERT - most powerful NLP algorithm by Google course on Udemy