Team 34: Civilized AI for Understanding Complex Q&A Content

Vivek Rathi
vorathi@ncsu.edu

Soumya Chintalapudi
schinta4@ncsu.edu

Trupti Sarje
trsarje@ncsu.edu

Karthika Vadivel
kavadive@ncsu.edu

I. Introduction and Background

In the recent years Natural Language Processing has witnessed its ImageNet moment. The Natural Language Processing (NLP) community has been putting forward incredibly powerful components that can be freely downloaded and used. Pre-trained models such as BERT have made NLP more accessible and accurate for custom language modelling tasks. This project is an attempt to built state of art Q&A labelling model using NLP technique namely BERT.

A. Problem Statement

Our proposed project idea stemmed from Google's Quest Q&A labeling. The motivation of choosing this is our interest to explore the realms of Natural Language Processing. Idea is rather simple where the AI model is trained to quantify the question and answer the way humans do. The challenge here is to make the model learn what comes naturally to humans, that is connection-based learning, and context-sensitive, mutual constraint, satisfaction bias. This attributes can summarized to contextual description of the given text. We aim to study the classical Bidirectional Encoder Representations from Transformer (BERT) model and use it to get context based vector input feature for final AI model to predict scores for the 30 attributes quantifying the Q&A understanding of a computer. Under implications of getting our baseline model and results we may try using other BERT based or related models for comparative analysis or either bagging methodology. Applying concepts learned in the class, we would perform data engineering and design a model considering the over-fitting under-fitting dilemma by tuning hyper parameters to get improved accuracy.

B. Related Work

For the purpose of Q&A AI would need 3 main components: language understanding, long-term contextual awareness, and a general knowledge-base. *'Semantic Search Machine'* is one Machine learning Method used in the past. The intrusion here is that a specific topic is more likely to contain some words compared to others. The classic example of this is that a topic about cats is more likely to contain words such as milk, meow, and kitten. While a topic about dogs is more likely to contain words such as puppy, bark, and bone. In the end, the topic mixture of a document should be the mixture that is most likely. Google has its own DeViSe algorithm to perform semantic modelling from visual and semantic information and there are many more such models such as the CodeSearchNet [1]. *'Similarity measure'* is yet another approach to this task. To compute the similarity between two documents, where one is the question asked by the user and the other being a reference question with a predefined answer, the cosine similarity measure is used.

With advances in deep learning, neural network variants are becoming the dominant architecture for many NLP tasks. Traditionally, most of the research in this domain used a pipeline of conventional linguistically-based NLP techniques, such as parsing, part-of-speech tagging and coreference resolution. Many of the state-of-the-art Q&A systems for example, IBM Watson use these methods [2]. However, with recent developments in deep learning, neural network models have shown promise for Q&A. Although these systems generally involve a smaller learning pipeline, they require a significant amount of training. GRU and LSTM units allow Recurrent Neural Networks (RNNs) to handle the longer texts required for QA. Further improvements such as attention mechanisms and memory networks allow the network to focus on the most relevant facts. Such networks provide the current state of art performance for deep learning based Q&A [3].

II. Method

A. Approach

BERT (Bidirectional Encoder Representations from Transformers) is a paper published by researchers at Google AI Language. It has caused a stir in the Machine Learning community by presenting state of the art results in a wide variety of NLP tasks.

BERT's key technical innovation is applying the bidirectional training of Transformer, a popular attention model, to language modelling [4] [5]. This is in contrast to previous efforts which looked at a text sequence either from left to right or combined left-to-right and right-to-left training. The paper's results show that a language model which is bidirectionally trained can have a deeper sense of language context and flow than single-direction language models [4] [5]. In the paper, the researchers detail a novel technique named Masked Language Model (MLM) which allows bidirectional training in models in which it was previously impossible. BERT is a pretrained model on a predetermined vocabulary set.

BERT is trained on combined corpus from BookCorpus (800M words) and English Wikipedia (2,500M words), ignoring lists, tables and headers. Each input is a two span text sampled from the combined corpus. BERT proposes a new pre-training objective called 'Masked Language Model' that randomly masks 15% tokens from an input sequence and trains on the model to predict the original vocabulary id of the masked word based on it context. This approach allows the representations to fuse context from either ends. Rather than always masking the chosen word, it masks the word 80% times with [MASK] token, 10% times with any random word and remaining 10% times with the actual word, thereby biasing the prediction towards the actual observed word.

In Question Answering tasks, the software receives a question regarding a text sequence and is required to mark the answer in the sequence. Using BERT, a QA model can be trained by learning two extra vectors that mark the beginning and the end of the answer. [4] [5]

There are two phases to using BERT: pre-training and fine-tuning. During pre-training, the model is trained on unlabeled data over different pre-training tasks. For fine-tuning, the BERT model is first initialized with the pre-trained parameters, and all of the parameters are fine-tuned using labeled data from the downstream tasks. Each downstream task has separate fine-tuned models, even though they are initialized with the same pre-trained parameters.

B. *Rationale*

The aim of the task is to improve automated understanding of complex question answer statement. Our approach intends to utilise a predictive model that is better suited to capture human level intuition. The data-set was obtained from Kaggle. In order to interpret a textual data by a machine language model, the textual information must be represented by numbers that is well suited to capture the features of the text. This vector must be designed in such a way that it represents the semantic relationships among the different words. This can be illustrated by the popular example which is King is to Man is same as Woman is to Queen.i.e., King-Man+Woman = Queen.

The 'FastText' Text Vector is a library of word vectors in about 157 different languages.On a similar note, we can also build our custom word2vec GloVe vector based on data corpus.

As it is an NLP task , the first method that comes to mind is the vanilla RNN. However, as RNN's are very slow to train and since they suffer from the problem of vanishing gradients for long sentences, we will not opt the approach. The LSTM, with its forget gate, input gate and output gate seems like a more reasonable approach to opt for.

For example if my question were to be 'What is the capital of India?' , the LSTM model can provide a direct answer which could be 'New Delhi'. However if my question was 'some city in India', then the model is unsure of understanding that the sentence is a question or a straightforward answer to the question.

This line of thought makes us want to capture the similarities among the words and what to retain through the LSTM model. However, we are already aware of another popular deep-learning model which is good at capturing the spatial similarities among the data. This makes us want to combine the LSTM and CNN models to better interpret and understand the question to provide the most effective answer.

However, sequential input of data to the LSTM and computation using both LSTM and CNN is very time consuming and sequential approach to deep learning is an inefficient usage of the modern day computational resources. In order to effectively be able to utilize parallelization while at the same time conserving the context among the words, we chose to opt for the 'Transformer model' in Machine Learning The Transformer model consists of the Encoder and Decoder component with an inbuilt Self-Attention layer coupled with a feed forward network in order to carry out this task [6] [7]. The Fig 1. diagrams shows the transformer architecture in which the sample word vectors given as input is 'Thinking' and 'Machines'. Researchers at Google AI Language have provided us with the Bidirectional Encoder Representations from Transformers also known popularly as BERT. Stacking many encoders we get BERT. They can capture not only meaning of words, but also the context.

III. Experiment

The resources required for the project are both RAM and GPU. The platform that we are using to access these resources is Google Colab with 10GB RAM and 12GB NVIDIA Tesla K80 GPU.

A. *Dataset*

The Kaggle Data [8], in the form of question-answer pairs, was collected by the Crowd Source team at Google from approximately 70 websites. It is tabulated and in a Comma Separated Values (.csv) format with each row having one question and its corresponding answer along with 30 extra features. Input data from kaggle has 6079 samples, with 41 feature columns. First 11 are to be used as input.

The input attributes or features are qa_id, question_title, question_body, question_user_name, question_user_page, answer, answer_user_name, answer_user_page, url, category and host. The last 30 are the target labels.

These labels are in the range of [0,1] and are continuous in nature, indicating that our aim is not binary prediction. Target labels' features are identified as questions or answers by a given prefix 'question_title' and/or 'question_body' and 'answer_'. Attributes like category have 5 unique values and host has 63 unique values. Attributes question_title, question_body, answer, category and host are used in our experiment. This input data was split into training and testing data with 4559 and 1520 samples respectively. This split was done such that 75% of original data was used for training and remaining 25% was used for testing.

The features are very descriptive in nature and tell various facts about the questions as well as answers. For instance, along with the question's title and body and 9 other features as input, 30 features such as category, interesting, fact-seeking, conversational are to be predicted as target labels. In the case of answers, features include answer body, its URL as input labels and relevance, satisfaction, plausibility, helpfulness as some of the target labels. Shown in Fig 2 is a snippet of the

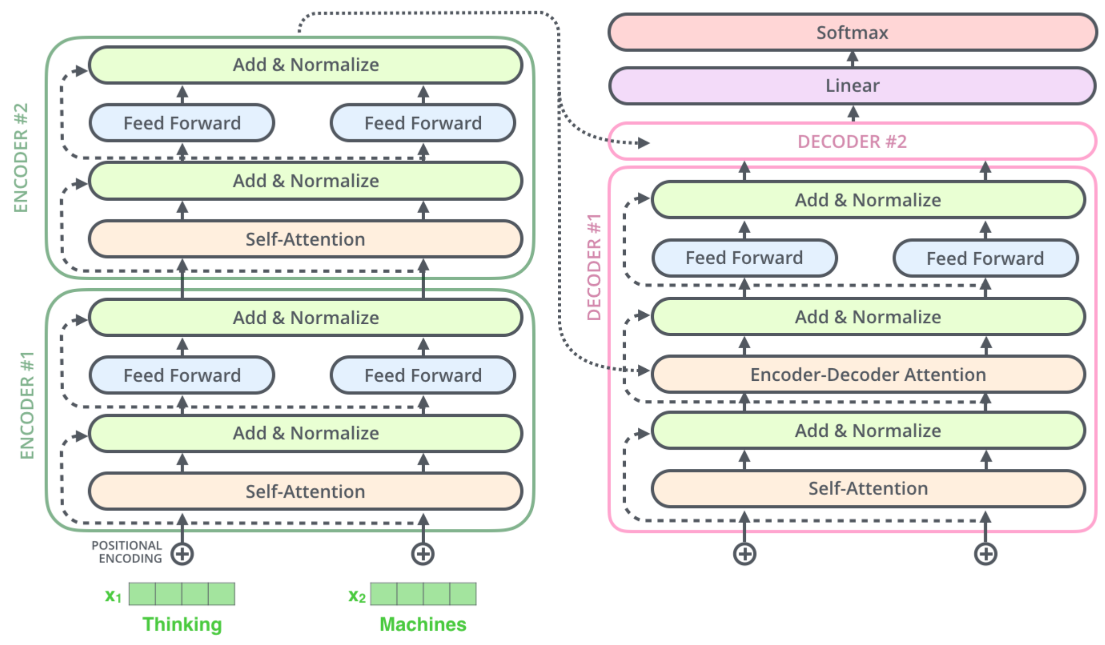


Fig. 1. The Transformer Architecture

training data-set [8].

B. Hypotheses

BERT model on its own has certain disadvantages as follows:

- Final Output should be less than a length of 512 since that is size of BERT.
- BERT was trained on vocabulary of 30,000 words which cannot be modified and would take a huge time to retrain. With the ever changing internet, if someone uses new internet Lingo, BERT cannot identify it.
- BERT model is prone to gender bias while predicting context based embedding such as "HE is a Doctor, SHE is a nurse / HE codes, SHE cooks".

The Linear regression model on the other hand, also has following short comings compared to Ridge regression:

- The linear regression model does not have a regularization factor while training the model which might result into abrupt weights.
- The linear regression model tend to perform bad on higher dimension data.

Due to these disadvantages and a pre-trained vanilla BERT model being used for feature extraction, we hypothesize that the final model performance can be improved by using a combination of PCA based reduced BERT features and Linear Ridge Regression.

This is because, PCA will obtain most significant features out of the BERT features. The regression model will be trained on reduced features which will make the model training better as the dependant variables are reduced. The ridge regression on the other hand has a regularization parameter that will keep the weights close to zero i.e, small weights. Hence, ridge regression will perform better than vanilla linear regression.

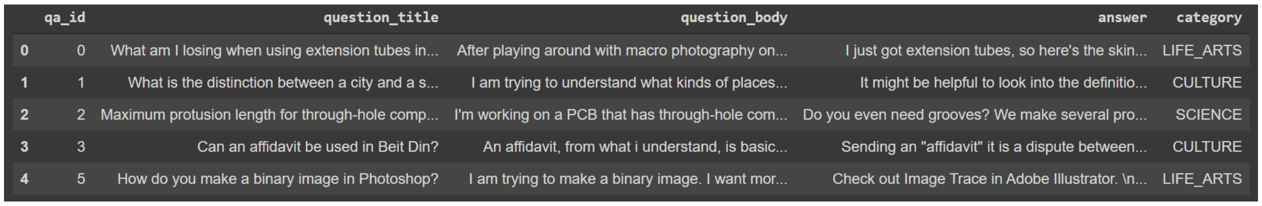
C. Experimental Design

The experimental setup can be visualized as two modules. The first module involves feature extraction. The second module is the Machine Learning algorithm which we use in order to get our output. From the dataset, we make use of three of the input columns as we found only those to be most relevant to the task. We have four different kinds of feature extraction techniques which we will use to make our input feature vectors to our Machine Learning Model.

- *Full BERT [F_BERT].*
- *PCA reduced BERT [R_BERT].*
- *One-hot-encoding with full BERT[O_F_BERT]*
- *One-hot-encoding with PCA reduced BERT [O_R_BERT]*

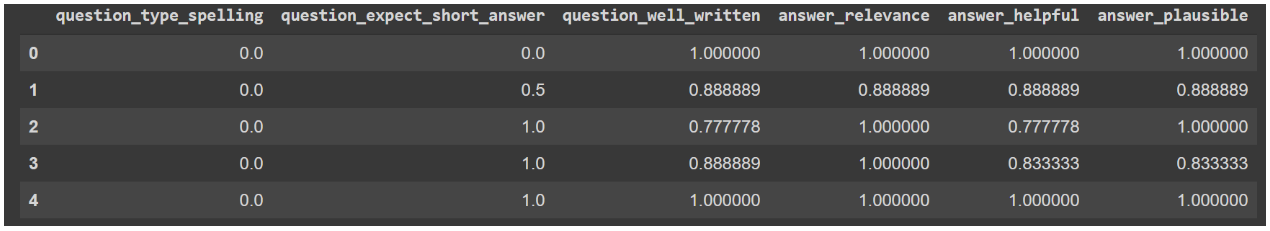
*1) **Vanilla BERT feature Extraction [F-BERT] :*** Only three of the 11 input columns were utilized from the dataset namely: question_title, question_body and answer. These were observed to be the most relevant of the input attributes for our task. The output that was finally generated was a 768 sized embedding vector for each attribute. In total there was 2304 output parameters for every sample.

BERT is a pretrained model trained on the English language with the perspective of Masked Language Modeling . The model is particularly successful at learning the context based nuances of the english Language. We are using the The BERT base Uncased model structure.The BERT model that we used is "BERT-Base, Uncased" [9]. It has 12 layers, 768 hidden units, 12 heads, 110M parameters. It is uncased, meaning all the text is lower cased before performing WordPiece Tokenization. (Fig:4) and a vocabulary size of 30,000. The inputs of the model are then of the form: "[CLS] Sentence Example [SEP]" The pre-trained model zip file contains a TensorFlow checkpoint with the pre-trained weights,



	qa_id	question_title	question_body	answer	category
0	0	What am I losing when using extension tubes in...	After playing around with macro photography on...	I just got extension tubes, so here's the skin...	LIFE_ARTS
1	1	What is the distinction between a city and a s...	I am trying to understand what kinds of places...	It might be helpful to look into the definitio...	CULTURE
2	2	Maximum protusion length for through-hole comp...	I'm working on a PCB that has through-hole com...	Do you even need grooves? We make several pro...	SCIENCE
3	3	Can an affidavit be used in Beit Din?	An affidavit, from what i understand, is basic...	Sending an "affidavit" it is a dispute between...	CULTURE
4	5	How do you make a binary image in Photoshop?	I am trying to make a binary image. I want mor...	Check out Image Trace in Adobe Illustrator. \n...	LIFE_ARTS

Fig. 2. Data snippet showing prefixed input labels of Q&A



	question_type_spelling	question_expect_short_answer	question_well_written	answer_relevance	answer_helpful	answer_plausible
0	0.0	0.0	1.000000	1.000000	1.000000	1.000000
1	0.0	0.5	0.888889	0.888889	0.888889	0.888889
2	0.0	1.0	0.777778	1.000000	0.777778	1.000000
3	0.0	1.0	0.888889	1.000000	0.833333	0.833333
4	0.0	1.0	1.000000	1.000000	1.000000	1.000000

Fig. 3. Data snippet showing target labels of Q&A

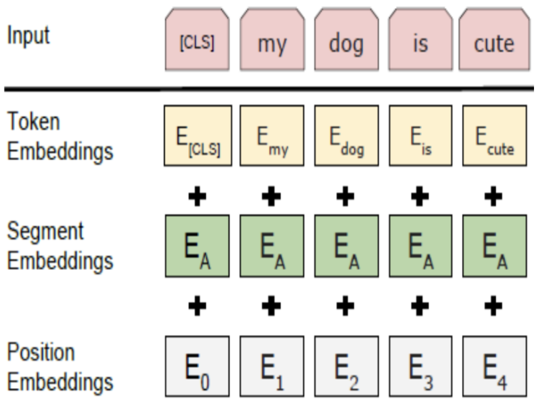


Fig. 4. WordPiece Embeddings used by BERT

[CLS]	101
team	2136
p	1052
##34	22022
is	2003
working	2551
on	2006
nl	17953
p	2361
bert	14324
.	1,012
[SEP]	102

Table I. Words and their corresponding vocabulary index

a vocabulary file for mapping WordPiece to word ID, and a configuration file containing the hyperparameters of the model. There are a couple of steps we make the pretrained BERT model perform:

- Tokenization: The original word has been split into smaller subwords and characters. Words that are not part of vocabulary are represented as subwords and characters.After breaking the text into tokens, we then convert the sentence from a list of strings to a list of vocabulary indices.
- Segment ID: BERT is trained on and expects sentence pairs, using 1s and 0s to distinguish between the two sentences. That is, for each token, we must specify which sentence it belongs to: sentence 0 (a series of 0s) or sentence 1 (a series of 1s). This is obtained from the BertTokenizer library.
- Extracting Embeddings: When we fetch we the hidden states of the network, Hidden states has four dimensions, in the following order:
  - The layer number (13 layers) : 13 because the first element is the input embeddings, the rest is the outputs of each of BERT's 12 layers.
  - The words / number of tokens
  - The batch number (1 sentence)
  - The hidden unit / feature number (768 features)

  For each token of our input we have 13 separate vectors each of length 768.We concatenate the layers, giving us a single word vector per token. The features obtained in such a way may be used in our ML model for prediction.

2) ***PCA reduced BERT [R_BERT] :*** The PCA reduction on BERT reduced the output embedding from 768 to 70 for each of the input attributes. The final output had 210 parameters for every sample. With the goal of reducing the contextual embedding dimensionality, we first processed our data using a pre-trained, uncased BERT Base model. Then, we compressed the data to a lower dimension using both PCA. This method aims to capture as much information as possible from the original BERT embeddings while preserving graphical locality information and nonlinearities in the final contextual embeddings. This can be visualized by plotting the features on a 2 dimensional PCA scale. In Fig: 5 as shown ,we can notice that questions regarding photography/camera are placed together. Similarly questions regarding cities are placed together and the others are placed at a distance apart. Since PCA preserves the variance in data while preserving contextual similarities, we found it most ideal to perform

our experiment with PCA. To compute linearly-reduced dimensionality embeddings, we used PCA to reduce the 768- dimensional BERT embeddings down to a number of components to 70.

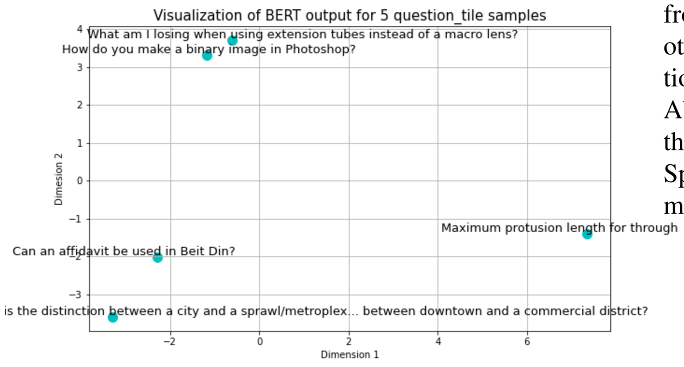


Fig. 5. PCA based visualization using eigen vectors

*3) **One-hot-encoding with full BERT [O_F_BERT] :*** The input feature vector now contained the one-hot-encoded representation of the 'host' and 'category' columns in addition to the length of the question_title, question_body and answer columns. There were 63 different hosts and 5 different categories and the performance was measured by incorporating these additional attributes as well. The final output has 2373 parameters for every sample.

*4) **One-hot-encoding with PCA applied BERT [O_R_BERT]:*** The input feature vector now contained the one-hot-encoded representation of the 'host' and 'category' columns in addition to the length of the question_title, question_body and answer columns. There were 63 different hosts and 5 different categories and the performance was measured by incorporating these additional attributes as well. PCA reduction to 70 dimensions was carried out. The final output has 279 parameters for every sample.

D. *Linear Regression*

Our aim is to predict the continuous values in range 0 to 1 for 30 output labels for each data sample. Linear regression is our chosen machine learning model. Regression model was trained on 4559 samples and tested on 1520 samples. Input feature space that is to be fed to regression model was experimented with 4 types of inputs mentioned above i.e. F_BERT, R_BERT, O_F_BERT and O_R_BERT. Initially Vanilla linear regression model was trained and evaluated using Mean Absolute Error (MAE) and Mean Squared Error (MSE) followed by Spearman Correlation Coefficient.

Regularized Linear regression with L2 regularization also known as Ridge regression was also performed on same set of inputs. Ridge regression was trained using 10 fold cross validation strategy with regularization parameter α ranging in [$1e^{-1}$, 1, 10, 100, 1000] with optimum α of 100 for models with input R_BERT and O_R_BERT whereas α was 1000 for inputs F_BERT and O_F_BERT.

IV. Results

A. *Results*

Vanilla linear regression and ridge regression models were trained on input feature space which was created from BERT model's output along with combination of other features as described in Experimental Design section. Error metrics like Mean Squared Error and Mean Absolute Error were used to evaluate and understand the efficacy of the linear regression models. Moreover, Spearman correlation coefficient was used to score the model with respect to y_test.

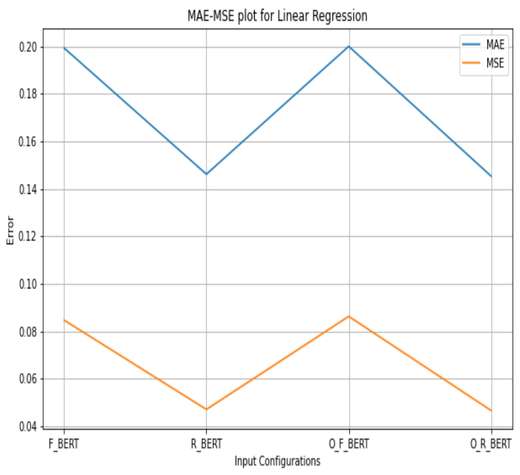


Fig. 6. MAE & MSE for Linear Regression

Fig: 6 shows the Mean Squared Error (MSE) and Mean Absolute Error (MAE) for Linear Regression model and Fig: 7 shows the same for Ridge Regression model. From Fig: 7, we see that O_R_BERT has lowest MSE of 0.0466 as well as MAE of 0.1453. Whereas Fig: 7 visualizing the MSE and MAE for Ridge Regression model disagrees with linear regression model. Ridge regression gives O_F_BERT with least MSE of 0.0453 and F_BERT with the least MAE of 0.1425. Hence, Ridge regression has a slightly better performance.

Spearman correlation score was computed for all input combinations. It evaluates the monotonic relationship between two continuous or ordinal variables, in our case, the continuous values of output predictor labels and their original values. In a monotonic relationship, the variables tend to change together, but not necessarily at a constant rate. The Spearman correlation coefficient is based on the ranked values for each variable rather than the raw data [10].

These correlation scores are summarized in table II and visualised in Fig: 8 and Fig: 9. Higher the Spearman correlation score, better is the model performance, meaning the predicted and actual labels are better correlated and have more similarity. Hence, referring to table II we can say that the model O_F_BERT has the best performance with the highest Spearman correlation score of 0.3602.

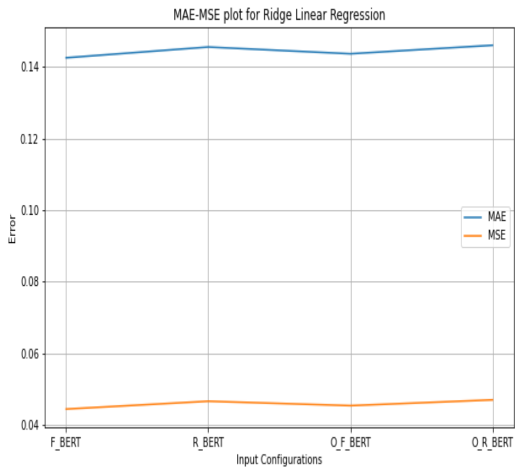


Fig. 7. MAE & MSE for Ridge Regression

Model	Input	Score
Linear Regression	F_BERT	0.2244
	R_BERT	0.3177
	O_F_BERT	0.2331
	O_R_BERT	0.3329
Ridge Regression	F_BERT	0.3516
	R_BERT	0.3231
	O_F_BERT	0.3602
	O_R_BERT	0.3377

Table II. Mean column-wise Spearman correlation coefficient

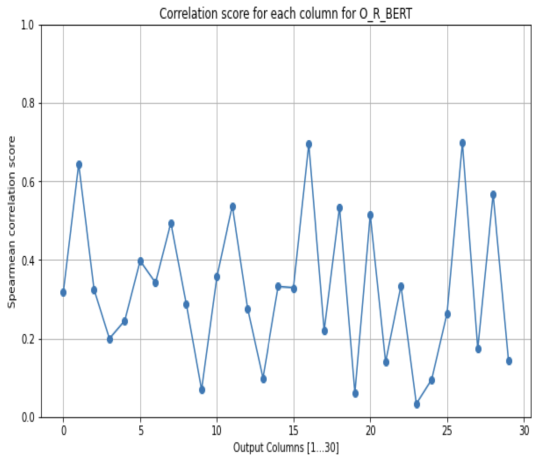


Fig. 8. Spearman Score for Linear Regression

B. Discussion

a) Result Interpretation and Justifications:

Spearman correlation coefficient or rank is used as a metric for two main reasons. First the Kaggle challenge responsible for the data used in the project had Spearman correlation score for the competition. Secondly owing to our continuous value based prediction in range of 0 to 1, it seemed ideal to verify if the predicted output is strongly or either linearly correlated with the ground truth.

This correlation is visualised for individual output label columns i.e. our 30 output labels as shown in Fig: 8 and

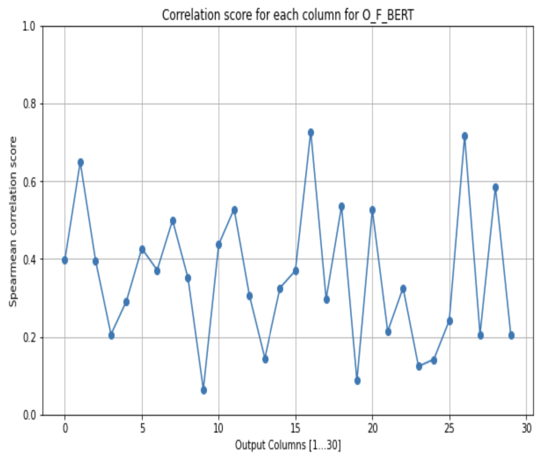


Fig. 9. Spearman Score for Ridge Regression

Fig: 9. Irrespective of the inputs, it can be seen from the figures that the highest 3 peaks and lowest 3 peaks are similar owing to their close correlation scores. Highest 3 peaks are nothing but the columns: question_body_critical, question_type_instructions and _type_instructions whereas lowest 3 are answer_plausible, question_type_spelling and question_not_really_a_question columns.

The small numbers of MAE and MSE give us impression that our model might be performing good. However, it is conspicuous that for linear regression model input O_R_BERT gives highest correlation coefficient, whereas for ridge regression model input O_F_BERT gives the highest correlation. These correlation values indicate the positive correlation with monotonically increasing function. Looking at the numbers we can conclude that inputs O_R_BERT and O_F_BERT perform best of their respective models.

We can infer that the input vectors from BERT model's output with combination of other non-BERT input features from original input data play crucial role in the prediction of linear and ridge regression model. Using ridge regression adds regularization and improves overall accuracy that is evident from MSE and MAE error metrics. Finally model with highest Spearman correlation coefficient performs good with the highest score of 0.3602.

b) Hypotheses Support:

As seen from the table II, the Spearman correlation for the model trained on all the BERT features with linear regression predictor had the worst performance. For the linear regression model the performance improved with reduced features by PCA as the dimension on dependant variable reduced.

On the other hand the overall performance with ridge regression improved but, the best performance was achieved with ridge regression trained on all BERT features and one hot encoding of question categories. As the improvement in performance by using "Category" feature improved by 0.0086 the "Category" did have some influence associated with it. It can be explained by referring to Fig: 10 as the "Technology" has most number of instances the model

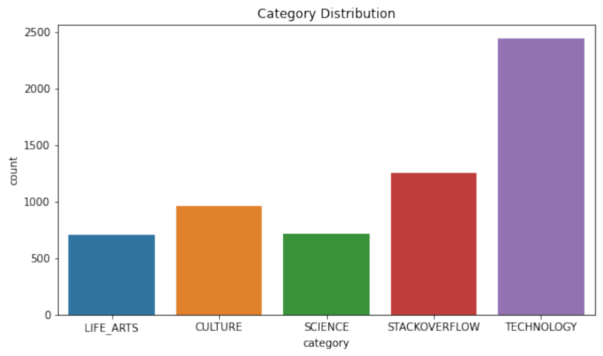


Fig. 10. Category Count

learned better to predict labels for ”Technology” category.

Hence, we can infer that our hypotheses was partially justified, with the BERT model giving a better performance with Ridge regression but not a good result with BERT’s reduced dimensionality. This could be because of the fact that ridge regression effects singular values due the regularization parameter and sets the weights close to zero. PCA also has similar effect where in it sets small singular values to zero. As Ridge has similar effect as PCA the model with full BERT feature and ridge performed better. Applying reduced features to ridge model would significantly lose data as dimensionality reduction will be performed two times.

c) Comparison with previous results:

The Spearman correlation score for our previous vanilla BERT model with Linear regression trained on all the features was 0.332. Since we worked on a combination model of one hot encoded BERT with Ridge regression, we got a better Spearman correlation score of 0.3602.

V. Conclusion

The study presented gives intuition on using pre-trained BERT model as a feature extractor to train a regression model designed to perform a specific task. It also presents advantages and disadvantages of BERT and possible ways to overcome the limitations.

As the BERT-Regression model works on ranking/scoring the Stack Overflow question and answers on the metrics such as ”Quality”, ”Relatability” and ”Correctness” the model could help improving the efficiency of search engine. Say, that a person is asking a Python query on Google, it will probably show display results for most relevant question on Stack Overflow, since it will have the best score for an evaluation metric such as Spearman Correlation.

Moreover, the model trained on human responses can be used to judge answer quality of an AI to produce more human like responses in applications such as virtual chatbots/assistants.

A disadvantage of our approach could be BERT’s limitation to just English language. Also, not being able to understand constantly emerging new terms/lingo that are not used for model training could possibly render an entire question moot and not interpretable by the search engine which would need constant training consuming time. Ensemble techniques can be a possible solution for these problems.

References

[1] A. Frome, G. Corrado, J. Shlens, S. Bengio, J. Dean, M. Ranzato, and T. Mikolov, “Devise: A deep visual-semantic embedding model,” in *NIPS*, 2013.
[2] J. W. Murdock, “This is watson. ibm journal of research and development 56,” 2012.
[3] J. Weston, A. Bordes, S. Chopra, A. M. Rush, B. van Merriënboer, A. Joulin, and T. Mikolov, “Towards ai-complete question answering: A set of prerequisite toy tasks,” 2015.
[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” 2019.
[5] J. Alammar. The illustrated bert, elmo, and co. (how nlp cracked transfer learning. [Online]. Available: http://jalammar.github.io/illustrated-bert/
[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” 2017.
[7] J. Alammar. The illustrated transformer. [Online]. Available: https://jalammar.github.io/illustrated-transformer/
[8] Google. Google quest q&a labeling. [Online]. Available: https://www.kaggle.com/c/google-quest-challenge/data
[9] G. jacobdevlin google. google-research/bert. [Online]. Available: https://github.com/google-research/bert
[10] Spearman correlation. [Online]. Available: https://support.minitab.com/en-us/minitab-express/1/help-and-how-to/modeling-statistics/regression/supporting-topics/basics/a-comparison-of-the-pearson-and-spearman-correlation-methods/#what-is-correlation