Industrial Engineering and Operations Research
COLUMBIA | ENGINEERING

Deep Learning for Human-Like Artificial Intelligence: Improving Automatic Question-Answering Algorithms

Sheila Teo

Master of Science in Business Analytics (Data Science)

Abstract

Question-answering is a discipline within the field of natural language processing that is concerned with building systems that automatically answer questions posed by humans in a natural language. To this end, the prowess of machines in answering objective questions with single, verifiable answers is undeniable. However, humans continue to still be better at addressing subjective questions that require a deeper, multidimensional understanding of context.

Context comes into play in question-answering in many aspects. For instance, questions can take many forms- some have multi-sentence elaborations as a fully developed problem, others may be one-liners arising from simple curiosity. They can also have multiple well-outlined intents or merely seek advice and opinions. It therefore becomes important to build algorithms that can understand the different subjective aspects of question-answering.

This paper utilizes publicly released data from Google Research to build predictive deep learning models for these different subjective areas of question-answering. As explained by the Google team, the demonstration that these subjective labels can be predicted reliably can "inform the way future intelligent question-answering systems are built and contribute to them becoming more human-like[1]."

Importantly, I demonstrate that specific deep learning architectures are indeed able to predict these subjective labels accurately. In addition, I explore multiple state-of-the-art word embedding models such as GloVe and BERT to investigate the significance of techniques chosen to transform text data in influencing the prediction power of deep learning models.

Dataset

The dataset used in this paper is obtained from the Google QUEST Q&A Labeling competition held on Kaggle in 2020. It consists of 10,000 question-answer pairs gathered from 70 different question and answer websites and cover 6 different categories. Examples of websites include Stack Overflow, Stack Exchange, and Ask Ubuntu while examples of categories include culture, science, and technology.

These question-answer pairs are shown to a group of human raters, who scored their quality in 30 different aspects, each on a scale of 0 to 1, based on their individual subjective interpretation of these quality score metrics. These 30 quality scores serve as target values to be predicted for each question-answer pair. A detailed list of these 30 targets can be found in Figure 1 in the Appendix.

[1] https://www.kaggle.com/c/google-quest-challenge/overview/description

Each row in the dataset contains a question-answer pair, with the following text features: question title, question body, and answer. The features used in this prediction task are thus these texts.

The segregation of this dataset into training and test sets utilizes a 90-10 split (10% of the dataset is reserved for testing). In particular, I choose to retain a smaller test set for this task due to the fact that the number of rows in the original dataset is small. As such, I allocate as many rows as possible for training in order to build a more robust deep learning model.

Exploratory Data Analysis

I start by exploring the data in order to uncover important patterns and relationships. I first investigate the categories that the question-answer pairs fall under using a bar graph in Figure 2, which depicts that the majority of question-answer pairs in the dataset fall under the 'Technology' category. As such, I next seek to better understand the text in the 'Technology' category, by building a word cloud of the top 100 words used in its questions and answers. Importantly, I remove all stop words and uninformative words before building this word cloud. For example, the word "wondering" is an uninformative yet frequently used word in questions posted online. This word cloud is depicted in Figure 3.

This prediction task is unique in that 30 targets are to be predicted. As such, it becomes important to investigate these 30 targets. Figure 4 shows a boxplot of all 30 targets, whose values range from 0 to 1. A significant observation follows- 10 of these targets take on less than 10 unique values each. This indicates that these 10 targets should instead be regarded as categorical instead of continuous, contrary to the initial impression that all 30 targets are continuous since they are scored on a continuous scale from 0 to 1. This discovery has crucial repercussions during subsequent model building.

Preprocessing of Text Features

In order to clean text data, I conduct the following:

- Convert all text to lowercase
- Separate joined words commonly used in speech. For instance, "I'm" is separated into "I am".
- Remove punctuation
- Remove stop words

Transformation of Text Features

Deep learning models require text features to be suitably transformed into a numeric representation beforehand. In the field of natural language processing, there are at present multiple state-of-the-art techniques for obtaining word embeddings, where individual words are represented as real-valued vectors in a predefined vector space. In order to determine the best technique for this specific dataset and prediction task, I attempt two different state-of-the-art methods and subsequently observe model performance on each.

1) Transfer Learning Using GloVe (Global Vectors for Word Representation)

One of the main design decisions when building a deep learning model with text data is the question of whether to utilize pre-trained word embeddings or to construct an embedding layer within the model to be trained from scratch with respect to the problem at hand. GloVe is one such pre-trained word embedding algorithm, trained on Wikipedia and Gigaword data. The advantage of GloVe lies in its utilization of both local statistics (local context information of words) and global statistics (word co-occurrence) to obtain word embeddings.

Since the dataset available for this task is relatively small with 10,000 rows, I choose to employ the transfer learning method using GloVe's pre-trained 100-dimensional word embeddings instead of training my own. In order to do so, I first tokenize and pad all text using Keras' Tokenizer class. Within the subsequent model architecture, I replace the parameters of the embedding layer with these pre-trained embeddings and freeze the layer, preventing its weights from being updated during model training.

Importantly, since the test set must remain unseen to avoid introducing bias, I utilize the same tokenizer trained on the training set in order to tokenize words in the test set. This implies that words found only in the test set and not the training set cannot be embedded, and have their embeddings set to 0. I also pad to the same length and specifications as the training set.

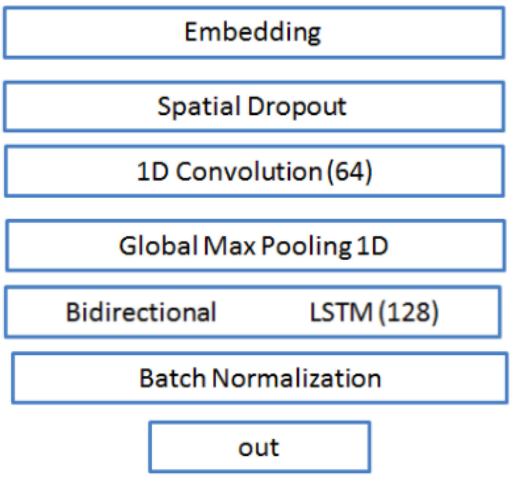
2) Utilizing BERT (Bidirectional Encoder Representations from Transformer) Embeddings as Features

BERT is a pre-trained language model created at Google that applies the bidirectional training of transformers to language modelling, allowing for a deeper sense of language context. Traditional single-direction language models such as Word2Vec produce a fixed representation for a word regardless of the context within which the word appears. In contrast, BERT's representations for words are dynamically informed by the words around them.

As a second method to obtain word embeddings, I follow a strategy proposed by BERT's original creators, as follows: I first pass the dataset's text data into the pre-trained BERT model and obtain the weights on its last 4 hidden layers. The summation of these weights is then used as input features for the subsequent deep learning model. These were conducted using PyTorch with the Hugging Face library, a popular library for implementing state-of-the-art transformers in Python.

Model Architecture: Bi-directional LSTM with 1D Convolution

There exists many different model architectures possible for this prediction task. In order to determine the optimal architecture, I took inspiration from the winners of similar competitions on Kaggle. In particular, I experiment with the model architecture used by the winner of a recent NLP competition held by Quora[2]:



[2] https://www.kaggle.com/c/quora-insincere-questions-classification/discussion/80568/

The reasoning behind utilizing the above model architecture is as follows:

Spatial Dropout: 20% Dropout
I utilize spatial dropout in order to overcome the problem with classical dropout methods, which occurs when adjacent input values are highly correlated.

1D Convolutional Layer: 64 Filters, Filter Size=3, Same Padding, ReLU Activation Function
The combination of both convolutional and LSTM layers is an increasingly popular model architecture in text modelling. The presence of both types of layers builds a network that is able to utilize both spatial and temporal information. Specifically, the convolutional layer extracts spatially-aware features from word embedding inputs and the subsequent LSTM layer interprets these features across time.

Global Max Pooling
Global pooling differs from traditional pooling in that it samples the entire feature map to a single value instead of down sampling patches of the input feature map. This prevents the model from becoming too complex in order to boost its generalization power.

Bi-directional LSTM: 128 Hidden Units, 20% Dropout
Given that this task involves text, a bi-directional LSTM is appropriate in order for the network to learn both backward and forward information at each step.

Batch Normalization
Batch normalization standardizes layers to produce a mean of zero and standard deviation of one. This allows for further model regularization.

Apart from model architecture, an important decision of choice is the loss function. Here, I implement the binary cross-entropy, due to two key reasons:

1. The 30 targets have a value of between 0 and 1. As such, they can be treated akin to probabilities under categorical target prediction. The usage of binary cross-entropy loss ensures that predictions for these target values remain between the range of 0 and 1, as compared to using traditional regression-based losses.

2. As described under the Exploratory Data Analysis section, investigations into the 30 targets depict that 10 of these targets should instead be regarded as categorical instead of continuous, further lending strength to utilizing binary cross-entropy loss instead of regression-based losses for this prediction task.

Model Tuning

Figure 5 depicts the cross-entropy loss obtained against number of epochs for both the training and test set using the above model architecture. We observe that severe overfitting occurs, since the loss on the training set and test set move in entirely different directions across epochs. I therefore start tuning the model architecture by increasing model regularization. Specifically, I increase traditional dropout from 20% to 40% and add recurrent dropout at 40% to the LSTM layer in order for the recurrent weights to be additionally regularized.

Re-running the model produces the new results depicted in Figure 6. It appears that regularization has helped to reduce model overfitting, since test loss and training loss are now far closer in value. However, a stark problem still exists- the minimum test set loss of 0.954 remains relatively high for this prediction task.

I therefore explore increasing model complexity next in order to allow the model to capture more signal within the data. Specifically, I do the following:

1) Increase number of convolutional layers from 1 to 2
2) Increase number of LSTM layers from 1 to 3
3) Furthermore, I tune the learning rate used under the Adam optimizer and find an optimal value of 0.04.

Results

The above final model design is run on both types of word embeddings obtained from the two state-of-the-art techniques described earlier. Figure 7 depicts the cross-entropy loss obtained using the first technique of GloVe embeddings, while Figure 8 uses the second technique of BERT embeddings.

BERT embeddings result in a minimum test set cross-entropy loss of 0.552, attained at epoch 159. In constrast, GloVe embeddings appear to perform better for this specific dataset and use-case, achieving a minimum loss of 0.401, attained at epoch 121. This minimum test loss is comparable to those obtained by others participating in this Kaggle Competition hosted by Google, thus validating that the final model architecture and hyperparameters above are indeed optimal for our prediction task.

Conclusion & Future Improvements

This paper explores utilizing a Bi-directional LSTM with 1D Convolutions to predict 30 different subjective areas of question-answering. The increased performance of the model after adjusting its design and hyperparameters serves to underscore the importance of model architecture in deep learning. However, one cannot overlook the significance of choosing the optimal techniques to transform text features into suitable numeric representations to be fed into the deep learning architecture. Given the buffet of word embedding techniques today, it is furthermore crucial to utilize multiple different methods in order to determine the one best suited for the specific dataset and task at hand, as demonstrated by this paper.

Tying back to the initial goal of this research, the purpose behind the release of this dataset by Google Research is to investigate whether the subjective and human-like aspects of question-answering can be reliably and accurately predicted by models. This paper validates that deep learning indeed is able to do so. Given this, the hope moving forward for the field of Natural Language Processing is that future intelligent question-answering systems can utilize similar deep learning models to produce more human-like output.

Appendix

Scoring Metric	**Explanation: To what extent… (Scale of 0 to 1)**
question_asker_intent_understanding	Can the question be understood
question_body_critical	Is the body of the question critical to understanding the question, apart from the question title
question_conversational	Is the question conversational in nature
question_expect_short_answer	Does the question expect a short answer
question_fact_seeking	Is the question fact seeking
question_has_commonly_accepted_answer	Does the question have a commonly accepted answer
question_interestingness_others	Is the question interesting to others
question_interestingness_self	Is the question interesting to the rater themselves
question_not_really_a_question	Is the question not really a question
question_multi_intent	Does the question have multiple intentions
question_opinion_seeking	Is the question opinion-seeking instead of fact-seeking
question_type_choice	Does the question provide choices
question_type_compare	Does the question require comparison
question_type_consequence	Does the question inquire about consequences
question_type_definition	Does the question inquire about definitions
question_type_entity	Does the question inquire about entities
question_type_instructions	Does the question inquire about instructions
question_type_procedure	Does the question inquire about procedures
question_type_reason_explanation	Does the question require an explanation
question_type_spelling	Is the question spelled correctly
question_well_written	Is the question well-written
answer_level_of_information	Does the answer provide sufficient information
answer_helpful	Is the answer helpful
answer_plausible	Is the answer plausible
answer_relevance	Is the answer relevant
answer_satisfaction	Is the answer satisfactory
answer_type_instructions	Does the answer provide instructions
answer_type_procedure	Does the answer provide procedures
answer_type_reason_explanation	Does the answer provide an explanation
answer_well_written	Is the answer well-written

Figure 1: 30 Quality Scoring Metrics (Targets in Prediction Task)

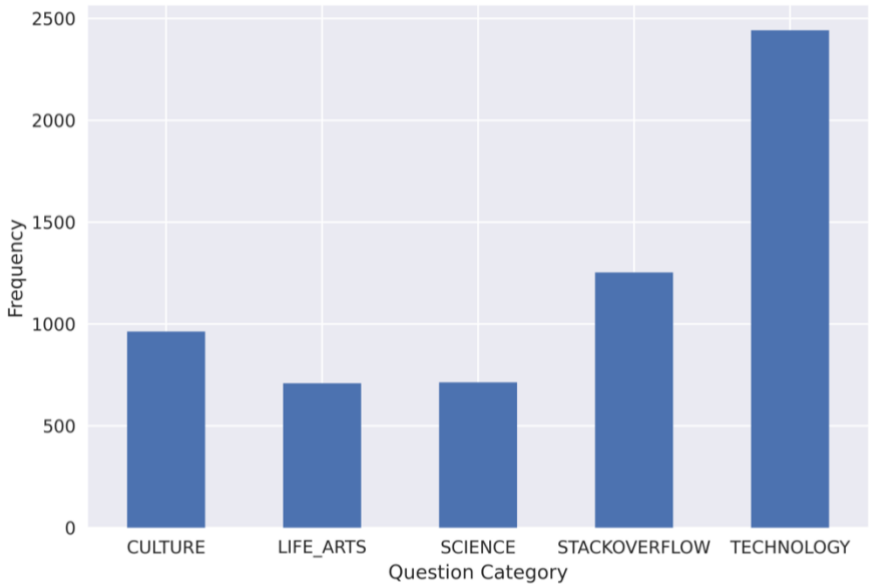


Figure 2: Frequency of Each Question Category

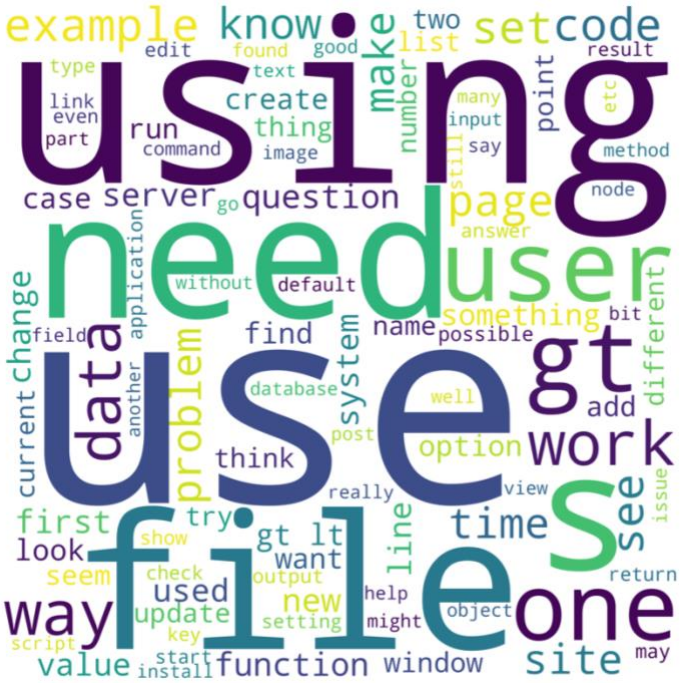


Figure 3: Word Cloud of Top 100 Words in Technology Q&A

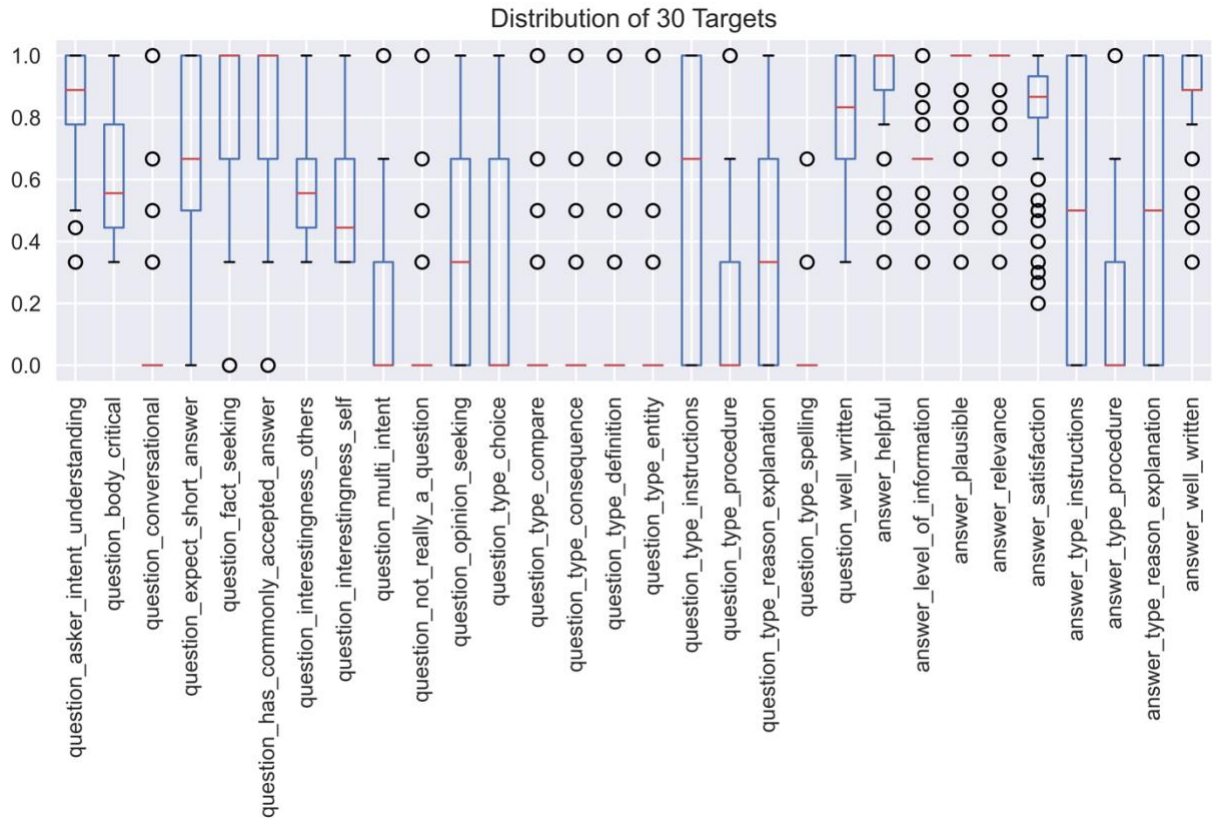


Figure 4: Distribution of 30 Targets

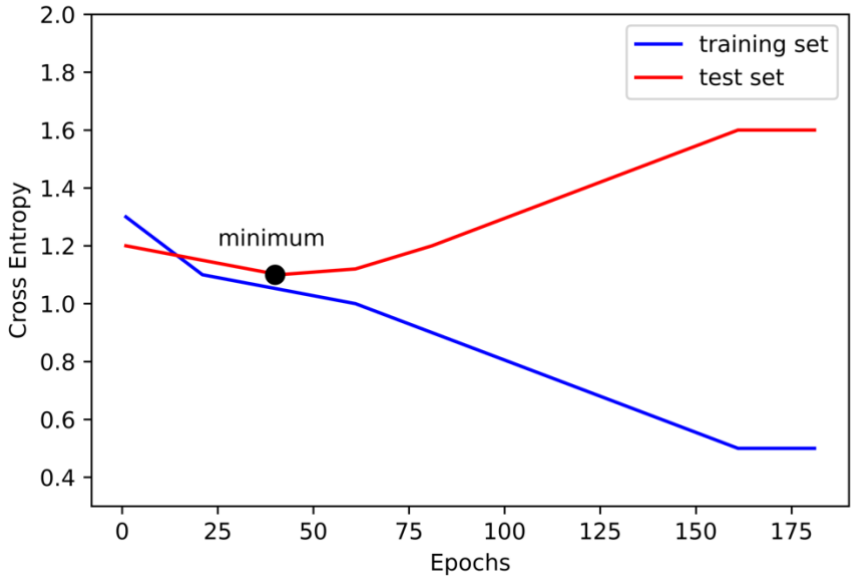


Figure 5: Pre-Tuning Results

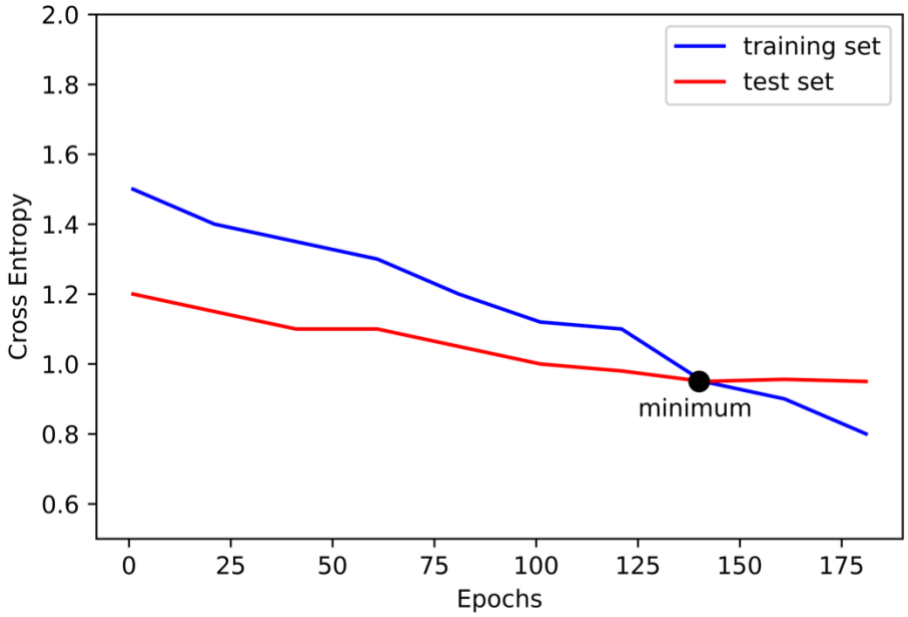


Figure 6: Results After Further Regularization

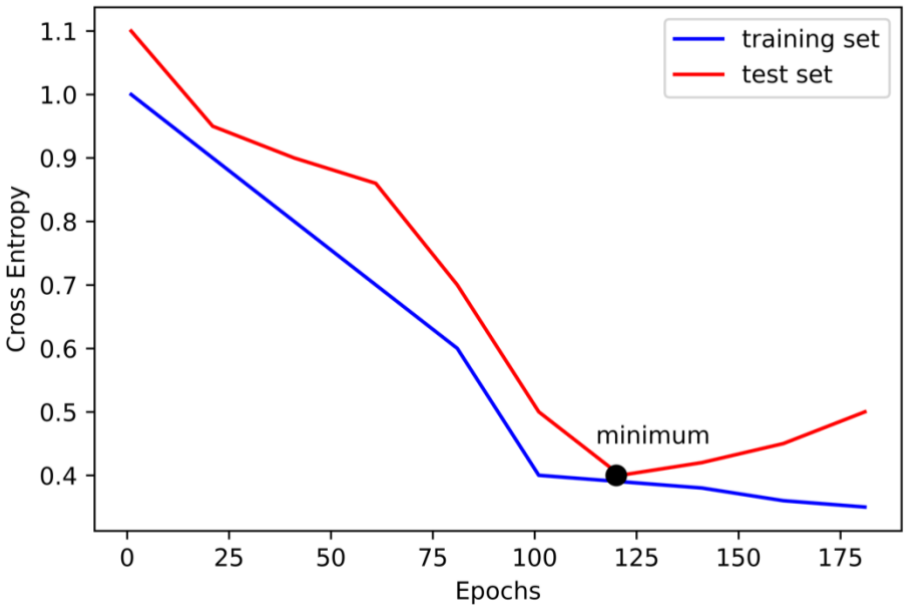


Figure 7: Final Model Using GloVe Embeddings

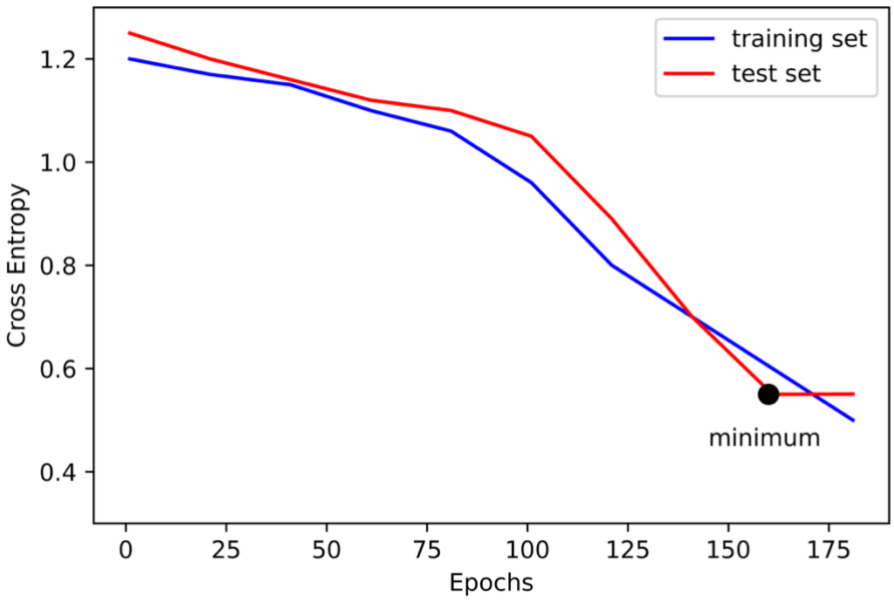


Figure 8: Final Model Using BERT Embeddings