CS542 Competition Report

The Autocast Competition

Akshaya Bali (U91195212)
Shivangi (U35642613)

1 Objective

We aimed to develop an accurate and calibrated machine-learning model for making forecasts. The questions utilized in the competition had been obtained from forecasting tournaments such as Metaculus, Good Judgment Open, and CSET Foretell, and typically require true/false or multiple-choice responses or predictions of numerical quantities or dates. The questions pertain to topics of wide public interest and have precise resolution criteria.

2 Introduction

In this report, we will talk about the different methods we used to implement an accurate machine-learning model for forecasting. We primarily used OpenAI's text-davinci-003 language model which is based on GPT-3 architecture. This model is available through OpenAI's API. Our goal was to develop a model that could accurately answer questions given some background information, and we employed different techniques to achieve this goal.

3 Approach

3.1 Just text-davinci-003

Firstly, we wanted to understand how the text-davinci-003 model works and that is why in our very first attempt to create a model for this competition, we simply implemented the text-davinci-003 model on the testing data. We fed the model the question, the background, and the choices and applied few-shot learning, where we asked the model the same question five times to improve its accuracy. We then modified the output to match the format expected by the competition organizers. However, this approach was not successful and did not give a good accuracy, but it did help us gain a better understanding of the task and gave insights to work on the model which is mentioned in approaches after this.

3.2 CountVectorizer with text-davinci-003

In our next approach, we wanted to use the training data to improve the accuracy of the model. To achieve this we decided to give the model some similar questions from the training set on top of the background information provided with the testing set question.

Now our new problem was to find a method to accurately find similar questions from the training set. To achieve this we first vectorized the texts using scikit-learn's count vectorizer and then applied cosine similarity to them. After this, we only saved the top three similar questions from the training set for every question in the test set. And, then ran the text-davinci model again, but this time, we also passed the question details of the top three similar questions from the training set. This approach improved our accuracy slightly.

3.3 BERT with text-davinci-003

For our final approach, we decided to use the BERT (Bidirectional Encoder Representations from Transformers) model for generating the embeddings of the texts. Specifically, we used the bert-base-uncased model, which is a pre-trained BERT model with 12 layers, 768 hidden units, and 110 million parameters. We also used the AutoTokenizer' tokenizer.batch_encode_plus method to tokenize and encode the text data with BERT, and the model was used to generate embeddings for the encoded tokens. Finally, we used cosine similarity to find the top three similar questions in the training set for each question in the test set. And, then similarly fed the text-davinci model as we did with the count vectorizer. This approach significantly improved the accuracy of our model.

4 Final Architecture

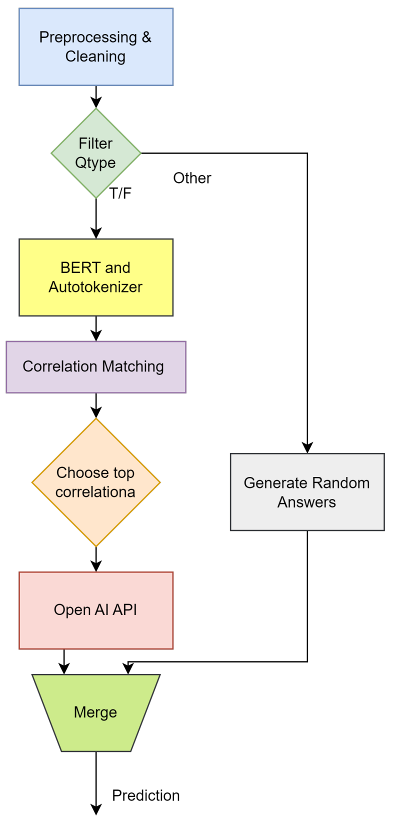
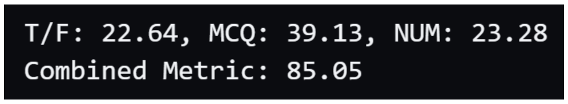


Figure 1: Architecture

From all the approaches we mentioned earlier, we tested an tried our model by segmenting the data into buckets using question type. We found out that OpenAI's API performs best for True or False types of questions rather than predicting the right date or choosing from a set of choices. This could be because the probability of a wrong answer is lowest for True False type of questions. *Figure 1* illustrates the final architecture we used for the competition.

5 Results



```
T/F: 22.64, MCQ: 39.13, NUM: 23.28
Combined Metric: 85.05
```

Figure 2: Final Model Accuracy

6 Conclusion

All in all, we realized that using pre-trained models like BERT with text-davinci-003 model and feeding similar data from the training set can significantly improve the accuracy of our machine learning models for forecasting. Our final approach with BERT embeddings and cosine similarity achieved the highest accuracy among the methods we used.