DD2417 - MiniProject QWP

Fredrik Nguyen

June 2, 2025

1 Introduction

Question answering (QA) systems have become a cornerstone of natural language processing (NLP), enabling machines to comprehend and retrieve information in response to user queries. While much attention has been devoted to improving QA accuracy, a closely related yet underexplored problem is Question Word Prediction (QWP), which is the task of inferring the missing question word or phrase when given a masked question and its corresponding answer.

For instance, consider the following QA pair:

Masked Question: <qw> is the capital of Sweden?

Answer: Stockholm

Here, the QWP task involves predicting the correct interrogative word "What" in place of the special masking token <qw> to complete the question. Unlike standard QA, which focuses on answer extraction, QWP requires reasoning about the inverse relationship: determining the appropriate question structure based on the expected answer. The masked token does not necessarily only be one question word, it can also be a *question phrase* such as "how many" and "how much" which increases the complexity of the problem.

The masked word in QWP tasks can quite often be easily be predicted by us humans, but how does a computer fare against the problem. In this project, we attempt to use a random forest model based on part-of-speech (POS) tags, and a deep learning model leveraging the BERT transformer architecture. The model that performed the best was the BERT model with a F1-score of 0.90 and accuracy of 90%. The random forest model achieved a F1-score of 0.51 and accuracy of 61% which barely beat baseline.

2 Background

Question words are words such as 'what', 'where' and 'how'. They signal that a sentence is interrogative and help to clarify the type of information being sought. For instance, if the information we are seeking is 'a thing' it is likely that we use the question word 'what'. If the the information instead is a relative position such as 'close to the thing', we might be using 'where' as the question

word. Part-of-speech tagging categorizes words into nouns, adjectives and verbs etc. based on its function, meaning, and usage within a sentence.

This is the inspiration for the first model, where we extract the information of the each word in the answer using POS in an attempt to represent the information being sought in the question. By using the part of speech tag as features we can use any classification methods to predict a question word. The architecture for the classification used was random forest. The random forest architecture is an ensemble method and predicts based on the aggregated results from many decision trees which in turn works by recursively splitting the input feature space into regions based on feature thresholds, creating a hierarchical structure of decisions that resemble a tree. This boost prediction ability and robustness compared to single decision tree.

The BERT, or Bidirectional Encoder Representations, model on the other hand uses contextual embeddings to understand the context of the question answer pair directly. BERT is built upon the Transformer architecture, specifically the encoder component with layers of Multi-Head Self-Attention, Feed-Forward Neural Networks, Layer Normalization and Residual Connections. The reason BERT was chosen was because of its pretraining on large amount of text with one of its objective being masked language modeling which is similar to the QWP task. Masked language modeling Randomly masks a small amount of the tokens in a sentence, and the model predicts the masked tokens based on bidirectional context. This enables BERT to learn rich, contextual word representations. The other objective is Next Sentence Prediction where the model predicts whether two sentences are consecutive. [1]

3 Methodology

3.1 Dataset

The dataset used for this project is the QA dataset sQuAD 1.1 by Standford university [1]. It consist of around 100000 question-answer pairs posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage as stated in the description of the dataset. In figure 1, we see some of the question answer pairs in the dataset. Most of the modeling is done on a separate dev-set before applied on the large main dataset.

3.2 Data processing

One of the major hurdles in the project was how the data should be prepared and masked. First of, the dataset did not contain information on the masking or even what the question word was supposed to be. There where also several questions in the data set that did not contain a question word at all e.g. those

[1] https://rajpurkar.github.io/SQuAD-explorer/

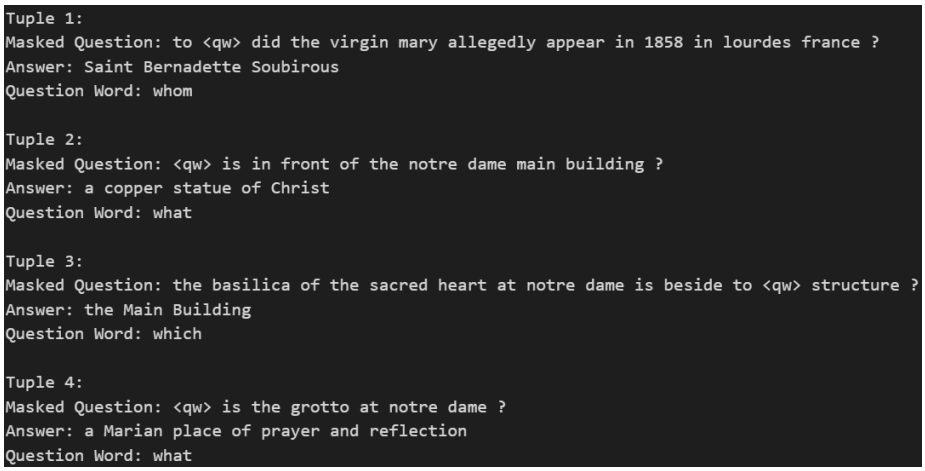

Figure 1: Some example of question posed in the dataset

where the answer to a question was to fill in a blank space, or question involving choosing between two alternatives.

We solve this by using the POS of the words in the question sentence as question words are interrogating words or wh-words, a class of English words used to introduce questions and relative clauses, which has special tags such as "WRB", "WDT", "WP" and "WP$". This solution also avoids problem arising from methods such as listing all possible question words and mapping them to the special token <qw> where sentences sometimes include several question words with different meaning depending on context. For example, 'who' is a question word in the sentence 'who are you?', but not in 'what is the name of the king who liberated Sweden from Denmark?'. The word 'who' would be masked in both cases if not for the use of POS which can distinguish both cases by assigning 'who' with different tags. We only mask words with the above tags for wh-words, and questions without them, we remove. However, these tags also including in certain cases non-question words such as 'that' and 'whichever' that we simply exclude.

To handle question phrases with 2 words e.g. 'how much' and 'how many' we look if the word following 'how' is an adverb or adjective with corresponding POS tags "RB" and "JJ". This method, however, gives in total 90 question words/phrases where the majority of them are "how" phrases such as 'how tall', 'how often' and 'how bad' etc. As many of them had very few samples, we decided on a cut off of at least 150 samples (somewhat arbitrarily) of the question phrase in the training set. This removed most of the question phrases and we end up with the following 12: 'where', 'how much','how long', 'when','what','which','how', 'whom', 'how many', 'who','whose','why'.

In fig. 2 we see that the data is heavily unbalanced with more than half of the sentences having 'what' as the target question word. We attempt to tackle

the problem by using weighted loss and also resampling equally of each class. However, it was found that both these methods gave worse results on the dev-set (around 84-87 % accuracy) compared to without balancing for Bert (90% accuracy). Hence, we did not do any balancing.

Finally, we split the data into 70:15:15 split for training, validation and testing for the BERT model and 85:15 for the random forest model as it does not require validation.

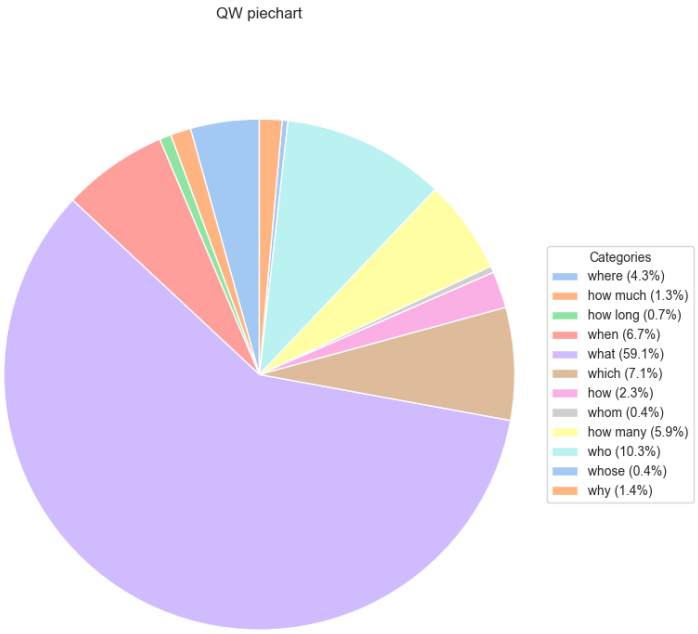

Figure 2: Distribution of the question words

3.3 Evaluation metrics

For evaluation metric we use accuracy and F1-score. We weight the F1-score based on the number of samples of each class. For instance, 'what' will have a lower contribution since it is a majority class, while smaller classes will have more impact. The evaluation is done on the test set.

For baseline we use "only guessing 'what'" as the data is very unbalanced and just guessing 'what' would give a accuracy of approximately 59% beating random choice.

3.4 Implementation details

We implement the BERT model using PyTorch. We use the tokenizer and the base pretrained model from hugging face python library including their implementation of the model. The BERT tokenizer embeds the words in the question answer pairs and concatenate them as the input the model. We keep the BERT model the same but include a dropout layer with 0.1 probability and a linear layer, with the BERT model hidden size as number of nodes, in the end for the classification task. For the finetuning on the dataset we use Adam optimizer with the learning rate 5e-5 which was found to be the best for finetuning in the original BERT [1]. For loss function we use cross-entropy loss. After each epoch we evaluate on the validation set and keep the model with lowest validation loss over 10 epochs. The model was trained on a A100 GPU with batch size 64 on training and validation and 1 in testing.

As for the random forest model, feature vectors where created using the POS tags by counting the number of times a tag would appear in the answer. The features was then fitted to the random forest model from python library scikit-learn with 1000 estimators and 30 max depth. The model was trained on CPU.

4 Results

As we can see in the table 1 we can achieve better result using both random forest method that make use of the POS tags and the BERT model that learn contextual embeddings through masking. Although, the random forest model barely beat the baseline in accuracy, the F1-score seems to be significantly better. On the other hand, the BERT models perform to great extent much better than both before mentioned models with 0.9 in both F1-score and accuracy.

Table 1: Performance Comparison of Random Forest and BERT

Model	Weighted F1 Score ↑	Accuracy ↑
Baseline	0.44	0.59
Random Forest	0.51	0.61
BERT	0.90	0.90

In figure 3 and 4, we can also see the confusion matrix for the prediction of both models. The BERT model seem to struggle with the word 'which' where it mostly predicts 'what' instead. This could possibly be attributed to the similarities in context of question answer pairs that use either of the question words mentioned. We note that there doesn't seem to be another case where the BERT models gets a lot of wrongs, which means it does not get affected by the skewed class distribution. This could be due to the massive amount of data that the model has pretrained on that makes the skew in finetuning insignificant.

As for the random forest method it seems to overfit due to the unbalance in data where it mostly predicts 'what'. However, if we disregard the rows and columns of the word 'what', we can observe in the diagonal where the values are the highest that the model seem to be able to correctly predict the labels in most cases. This this method is indeed working but it has some the drawback of being easily biased towards the majority class 'what' due to the unbalanced dataset.

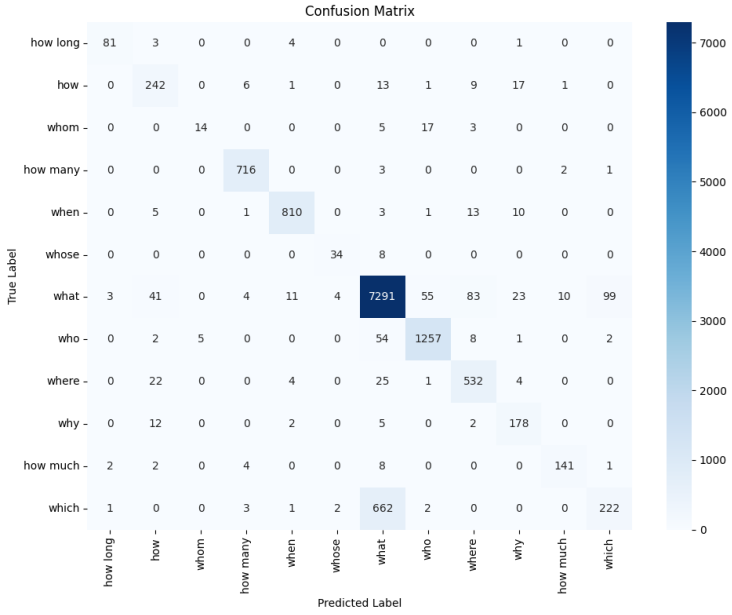

Figure 3: Confusion matrix for BERT

5 Conclusions

To summarize, we seem to be able to use machine learning with 2 very distinct methods with good results. A big hurdle was the unbalanced dataset, where a large number of question word is 'what'. However, this reflects reality in the use of question words where 'what' appears to be most popular such word. The random forest method seems to be biased as a result of it. One solution to this could be the of foundation model such as BERT that has pretrained on a lot of data, as it did not seem to be affected by it that much. Some future directions would be to explore the balancing of the dataset more including synthetic data generation of questions, as well as include other features into the random forest model e.g. named-entity-recognition. Either way both models beat the baseline, where the BERT model does it with wide margin.

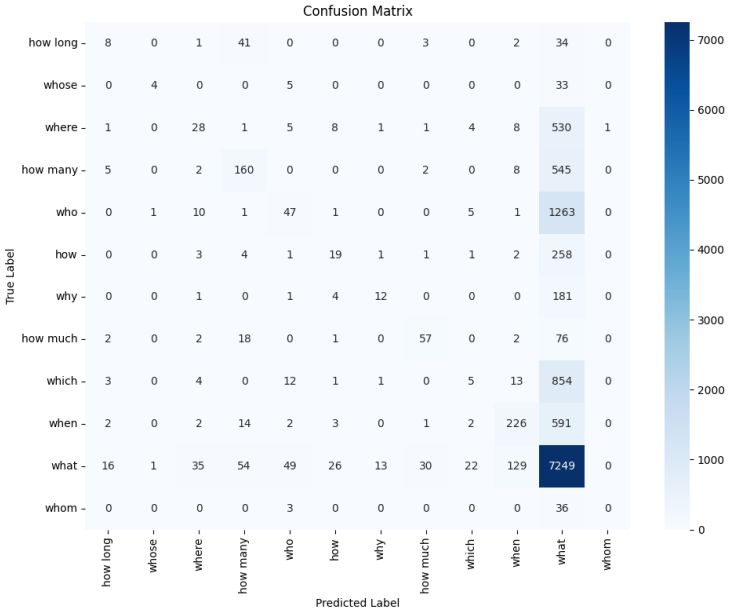

Figure 4: Confusion matrix for Random Forest

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.