CS 388 NLP Final Project Report: Attention Is All You Need With Question Answering Verification

Christopher Lawson
Department of Computer Science
The University of Texas at Austin

Rishi Salem
Department of Computer Science
The University of Texas at Austin

1 Abstract

Question-Answering models are good at generating answers when given the relevant context, but they can perform poorly when asked a question they lack the information to answer. Creating models that can identify and choose not to answer non-answerable questions is critical to reducing misinformation and creating truly helpful models. In this paper, we will discuss our implementation of the read-verify model, using verifiers built with transformers rather than LSTMs. We will explore the results of our work by showcasing our results, and discuss other steps that could be taken to improve the accuracy. Our new implementation was not able to get the same overall accuracies as the base, no verifier, models ($\sim \%90$), but it was able to increase the non-answerable accuracy to $\%70$ (from a base non-answerable accuracy of $\%0$) while having an overall accuracy of also $\%70$.

2 Introduction

2.1 Problem Definition

A question-answering model (QA model) is a model which, when provided a context and a question, provides an answer to the question from the context. However, no context provides enough information to answer all questions. Questions that cannot be answered with the provided information are called non-answerable questions. Ensuring QA models answer questions properly, even in the face of non-answerable questions, is a difficult problem. While it would be ideal to have models answer such questions by providing no response, or providing a response that makes it clear that there is no answer, this does not always happen; models may hallucinate answers or confidently provide misinformation (as shown in **Figure 1**).

This is clearly a problem, as language models should not perpetuate misinformation.

2.2 Proposed Solution

One solution to this problem is the reader/verifier model. This model, initially proposed by Hu et al Hu et al. (2018), is made of two parts. The reader is a typical question answering model; the verifier is an additional model that is trained to predict whether a question is answerable by providing the

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

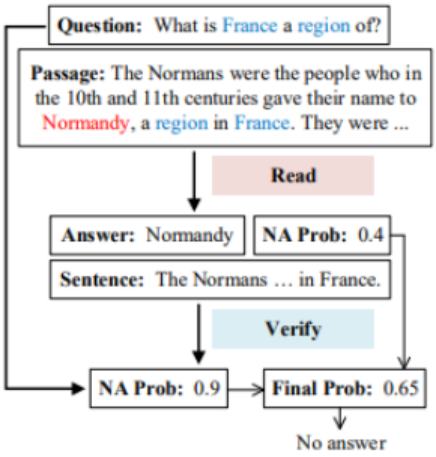

Figure 1: Example of Non answerable Question

question, the context, and the answer predicted by the reader. The general architecture is shown in **Figure 1**.

In this project, we created a read-verify model inspired by the model described in Hu et al Hu et al. (2018). We performed tests with two readers, an ALBERT model and a RoBERTa model fine-tuned on SQuAD 2.0. We also created two verifiers from scratch, one of which was directly taken from Hu et al. and one of which was adapted to use more modern technologies (transformers instead of LSTMs, view architecture in **Figure 4**).

In this paper, we will first discuss the processes used to create the readers and verifier. We will then explore the results obtained using the reader without the verifier, the results of each of the verifier models combined with each of the reader models, and the performance of the unified models on an erasure dataset.

3 Background

3.1 Related Works

As mentioned previously, Hu et al Hu et al. (2018). released an early paper on the answer-verifier model. This model was state of the art at the time because it could use any QA model to generate answers, and any suitable verifier could verify the answer; this ensures that answerable questions can be answered with the same quality as the existing QA model, and presumably only non-answerable questions will be affected by the verifier. The model described in the paper performed slightly better than a basic reader model on the SQuAD 2.0 dataset, a contrast set for SQuAD which contains non-answerable questions (for more information, see the "Datasets" section of this paper).

A newer report by Alfarizy and Mandala (2022) now takes advantage of BERT with a new architecture to also propose a similar architecture to Hu et al. (2018) however, the approach that they take for the verifier was to use BERT to embed the generated answer sentence and do sentence similarity (specifically through cosine similarity) with the target sentence. It also incorporates the non-answer probability by looking at the probabilities of the generated sentence to see how likely the sentence is to actually generate. This paper inspired us to use variations of BERT for question answering with unanswerable questions.

3.2 System Components

3.3 SQuAD 2.0

The dataset we used for training and testing our model was the SQuAD 2.0 dataset Rajpurkar et al. (2018), found here. This is a modified version of SQuAD containing non-answerable questions; it contains 100,000 questions from the original dataset and additionally contains 50,000 unanswerable questions created by crowd-workers.

Table 1: Reader only base results

Model - Dataset	Has Answer - F1	No Answer - F1
RoBERTa - Blank	91.27	0.00
RoBERTa - No Answer	91.63	0.07
ALBERT - Blank	89.48	0.00
ALBERT - No Answer	89.42	0.12

3.4 ALBERT and RoBERTa

The original read/verify model by Hu et al Hu et al. (2018), used the Reinforced Mnemonic Reader (RMR) Model Hu et al. (2017) to perform the question answering task. The RMR model was previously created by the authors of the read+verify paper for general-purpose reading comprehension tasks, and a codebase was not provided for the reader; as such, we felt that reimplementing the reader was outside the scope of the project as we wanted to try to focus on implementing the verifier side of this project. We also wanted to see if more modern question answering models could perform a better job of answering questions than the RMR model.

For the modern question answering models, we used ALBERT and RoBERTa models fine-tuned on the SQuAD 2.0 dataset as our reader models. Given that BERT models are the most state of the art models that we have access to off-the-shelf, we wanted to get some variations of them for our model. We ended up chosing two different variations: ALBERT and RoBERTa. ALBERT Lan et al. (2019) is a lighter version of BERT contains fewer parameters and requires fewer resources for training than comparable BERT models while still getting similar results to the original BERT. RoBERTa Liu et al. (2019) is a heavyweight, optimized version of BERT which boasts better performance than the original. We chose to train on both to determine whether RoBERTa's robustness would result in significantly higher performance than ALBERT while seeing if they could outperform the original papers Hu et al. (2018) RMR reader model.

4 Reader Model

4.1 Initial Issues

Our first step was to create the reader models. This caused us more trouble than expected, as the metric used when training on SQuAD 2.0 requires a probability of answerabililty (essentially asking the probability of a given question being answerable) which would be the output of a verifier we are constructing. Thus we had to stick with using a SQuAD 1.0 metric to train our reader model. This then led to the issue of the SQuAD 1.0 metric not being able to handle non-answerable questions as they were being represented by $null$ in the SQuAD 2.0 dataset.

We fixed this by replacing all null-answers with non-null answers. We chose two formats for non-answers; the first was an empty string "", and the second was the string "NO ANSWER". We tested both formats to see if one would perform better than the other; the results are shown in **Table 1**.

Here we see that both models are able to get around 90% on answerable questions regardless of the $null$ replacement strategy. However, we see that the models performed horribly on non-answerable questions.

One interesting thing that we do see is that the text that we replace the null value did have an impact on the F1 score. The table suggesting that replacing the $null$s with "No Answer" led to higher F1 scores. While not significant, it did spark our curiosity as to how including a verifier model would be affected. Thus, throughout our project, we had 2 datasets; one where the nulls were replaced with an empty string (the blank dataset) and one where they were replaced with the text "NO ANSWER" (the No Answer dataset).

4.2 Discussion

One thing that we did realize is that the increase in F1 scores from a blank answer to the "No Answer" questions was due to the way the score was calculated. The current metric that we're using to evaluate

the model performance is called "SQuAD" which was bundled with the dataset of the same name. Because this metric acts similarly to the BLEU score, the increase in F1 score on the "No Answer" dataset is potentially because the model could have predicted an answer that started with an uppercase "N" or have matched a few other letters to potentially drive up the score.

Thus, there may not be a true impact on the model performance and instead a hallucination from the evaluation metric. However, we continued using both datasets with the verifier model to determine whether any difference in performance would persist.

Continuing forward we will experimenting with 4 different Reader models, ALBERT trained with both "Blank" and "No Answer" datasets and RoBERTa trained with both "Blank" and "No Answer" datasets.

5 Verifier Implementation

Our first aim was to replicate the work to then be able to understand what was going on. As described in Hu et al. (2018), there are two parts to the overall model, the Reader part and the Verifier part. As discussed in the “Background” section, we used ALBERT and RoBERTA as our base Reader Models.

The verifier models we created was inspired by Hu et al. (2018). This paper describes three verifier models, in this project we tried to reproduce the first 2 models.

5.1 Model 1

The first model starts by concatenating all the inputs to the verifier into a single input. These inputs include the context (described in the figure above as the Answer Sentence), the question, and the answer predicted by the reader model. The architecture is shown in **Figure 2**.

Once these inputs are concatenated, they are sent through a basic classifier model consisting of a text and position embedding layer, a transformer with masked multi-head self-attention, a linear layer, and a softmax layer. The classifier was trained using a standard cross-entropy objective.

As mentioned previously, the infrastructure of this model was taken from the design described in Hu et al Hu et al. (2018). However, since no code was provided alongside the paper, there were parts in the implementation described in the original paper which were not clear. As such, we made some assumptions regarding the finer details of the implementation.

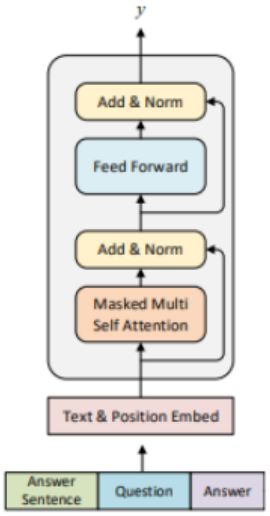

Figure 2: Model 1 architecture from Hu et al. (2018)

In particular, the paper describes the concatenation of several inputs to create a single input vector X, along with a delimiter token between the question and the answer; however, we encountered

difficulties in running the model with the delimiter token, so we removed it from the input. Additionally, while several details were provided about the transformer implementation, we assumed these were typical for any transformer implementation and used the Pytorch transformer encoder implementation.

The purpose of this Model 1 was to provide us a standard base line for us to compare our next model, Model 2, against while being on even footing. In the sense that, because making quite a view deviations from the original paper (due to code specific implementations and Reader model used), we needed a model to fairly judge whether the modifications we make to Model 2 are helpful or not.

5.2 Model 2

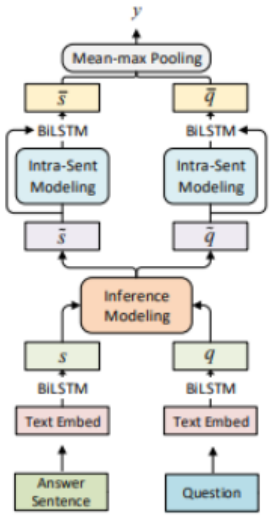

Figure 3: Original model 2 architecture from Hu et al. (2018)

The second verifier model focuses on the interactions between the sentence which contains the answer extracted from the Reader model and the question to determine answerability. The only inputs to this model are the answer sentence and the question. The original model in He et alHu et al. (2018) is outlined by **Figure 3**. The modification we want to propose is showcased by **Figure 4**. Where we want to replace the BiLSTM layers throughout the model with Transformer layers. Given what we learned in class about how Transformers have been outperforming LSTM and BiLSTM architectures we wanted to see if these changes would make a difference.

A more detailed explanation of the proposed Model 2 is as follows. For each input, the model creates word- and character-level embeddings, concatenates the two embeddings, and puts it through a transformer. (Both embedded inputs are sent through the same transformer; this is the weight-sharing that allows for identifying interactions between context and question.) The outputs generated by the transformer, when given the embedded context and question, are called the s and q vectors respectively.

The result is fed through the inference modeling layer. This layer creates a matrix of attention weights by computing the dot products of all s and q pairs, then normalizes the result. The normalized result is then run through a heuristic function involving a gelu operation, a sigmoid function, and linear layers.

Once inference modeling is complete, the results are run through another transformer, then added to the residual and run through a mean-max pooling layer. Finally, the result is run through a linear layer and a softmax layer to generate the probability that the question is non-answerable.

As with model 1, we implemented model 2 from scratch. As such, there were elements described in the original paper which were not kept in our implementation (beyond the obvious LSTM to transformer change). For example, the paper used GloVe for word embeddings, but did not specify the process for character embeddings; as such, we chose to train an embedding layer to perform both

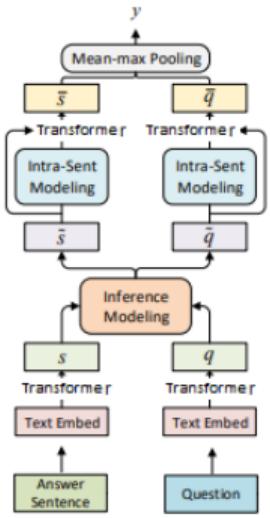

Figure 4: Proposed model 2 architecture

word and character embeddings using a vocabulary generated from SQuAD 2.0. Additionally, the word embedding (as described in the paper) would include a binary feature indicating whether the word was part of the answer; as SQuAD 2.0 can allow for several answers to the same question, we chose not to implement this binary feature.

5.3 Disclaimer

While we implemented the verifiers to the best of our abilities using the descriptions in the paper, we found that they did not perform as well as expected (as to be shown later). This may be caused by issues in our implementation, or may be because BiLSTMs provide capabilities that transformers cannot capture. In either case, we will discuss the results we collected with the assumption that the code was correctly implemented and that issues in performance were caused by the changes we made to the design.

6 Results

6.1 Model 1

Figures 5 through **Figures 8** contain the confusion matrices for the results gathered on the respective reader models and datasets given Model 1 as the verifier.

6.2 Model 1 Discussion

Here there are three elements that we would like to analyze: the overall accuracy of the 4 different models, the True Positive Rate - TPR (correctly classifying Answerable questions) and the True Negative Rate - TNR (correctly classifying non-answerable questions).

Looking at the first element, we see that our Model 1 across all different models is performing very poorly. Most of these models are performing as well, if not worse, than randomly guessing. This was very concerning given this could mean that the model was not able to learn anything from the encodings that we passed into it. However we can view this model in a slightly better light when looking at the TPR and the TNR for these different models. This silver lining being that we now have much better rate of answering Non-Answerable questions. Coming back into reality however we note that this increase in TNR comes at the cost of no longer having a very high accuracy of answering Answerable questions as shown in **Table 1**. But looking at **Figure 6** we see that it is encouraging to see that there is a balanced TPR and TNR.

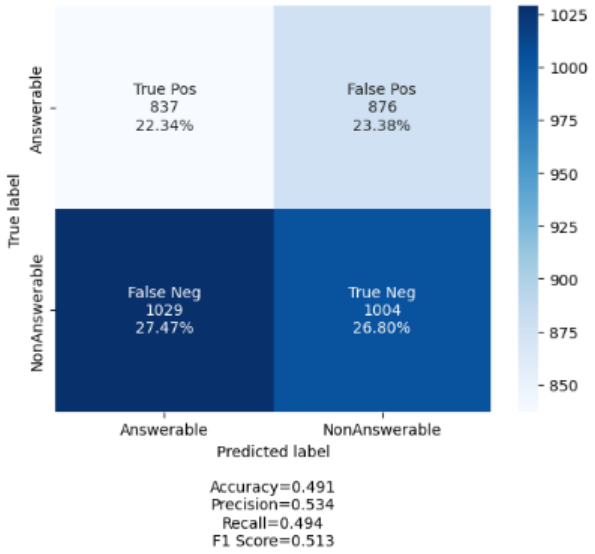

Figure 5: Confusion Matrix for Model 1 from ALBERT - Blank

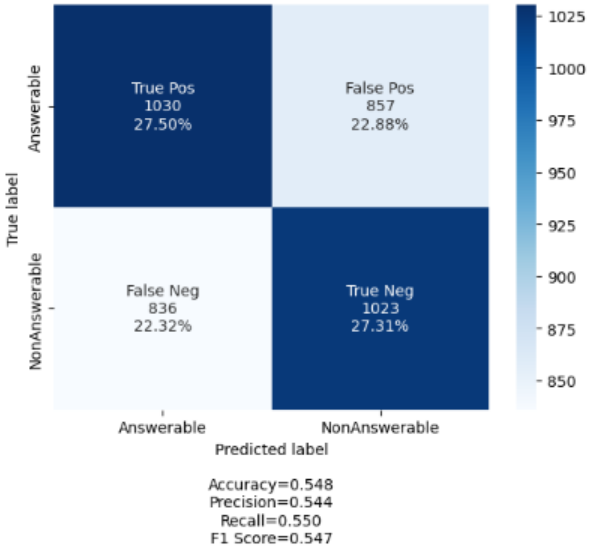

Figure 6: Confusion Matrix for Model 1 from ALBERT - No Answer

Two other elements that we wanted to focus on was the difference in performance across the "Blank" and "No Answer" datasets and across the RoBERTA and ALBERT models.

Focusing first on the difference between RoBERTA and ALBERT, we are not able to it did not seem to be that much of a difference in either the overall accuracy, TPR or TNR.

In regards to looking at the difference between the "Blank" and "No Answer" datasets, we see that TPR increases by roughly $\%10$ but the TNR decreases by roughly the same amount. At least when looking at model 1, it would seem that the "Blank" dataset is better when it comes to correctly predicting non-answerable questions but the "No Answer" dataset is better when it comes to correctly predicting Answerable questions. Thus going forward into the Model 2 results, we will look out to see if this pattern continues.

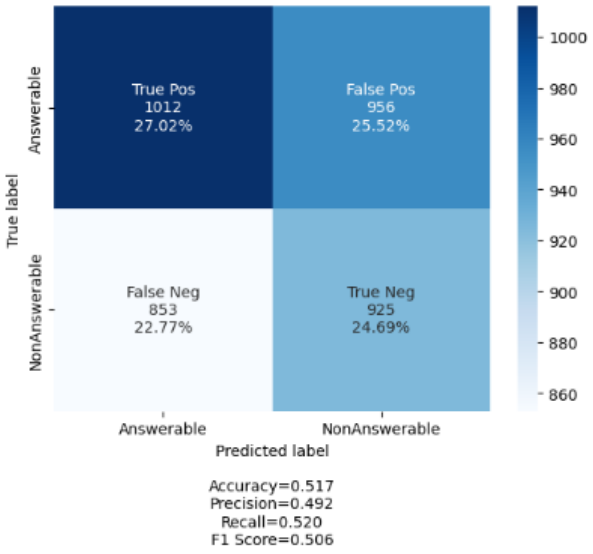

Figure 7: Confusion Matrix for Model 1 from RoBERTa - Blank

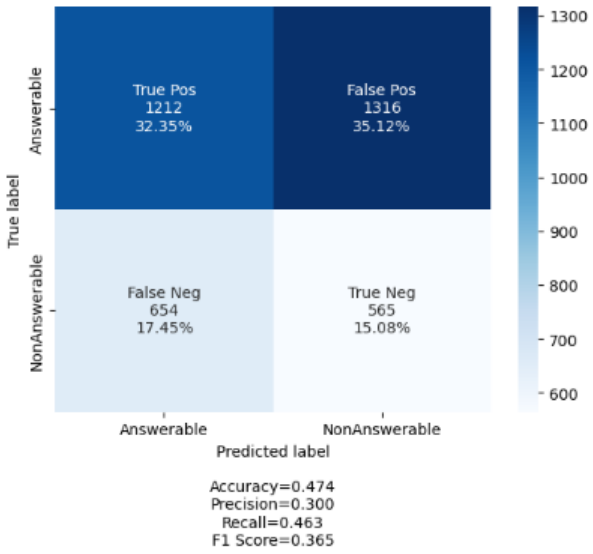

Figure 8: Confusion Matrix for Model 1 from RoBERTa - No Answer

6.3 Model 2

Figures 9 through **Figures 12** contain the confusion matrices for the results gathered on the respective reader models and datasets given Model 2 as the verifier.

6.4 Model 2 Discussion

Just like with the Model 1 Discussion, we will be looking at three elements that we would like to analyze: the overall accuracy of the 4 different models, the True Positive Rate and the True Negative Rate.

Comparing the results to Model 1, each of the 4 models performed better than their counter parts which is very encouraging. Again we are not quite at the level of the original baseline outlined in **Table 1** but we are almost there after viewing the results from the RoBERTa - Blank in **Figure**

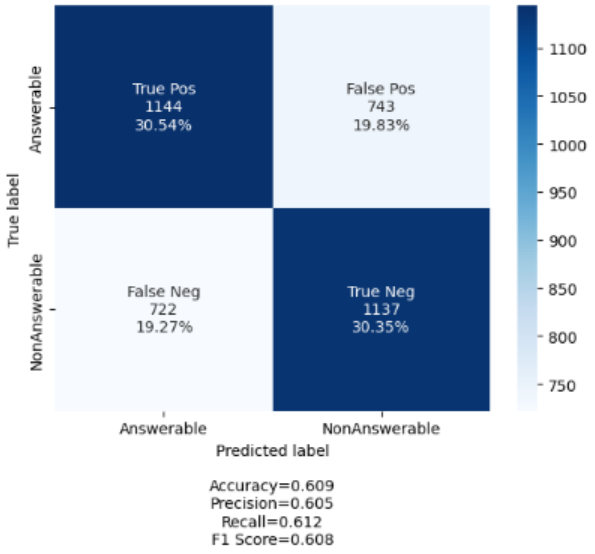

Figure 9: Confusion Matrix for Model 2 from ALBERT - Blank

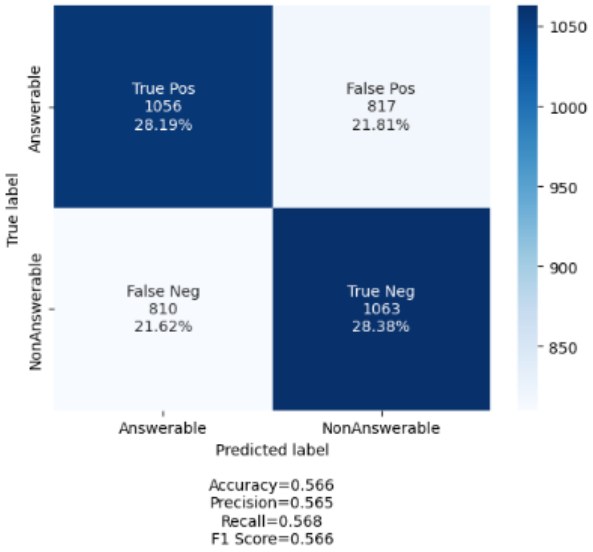

Figure 10: Confusion Matrix for Model 2 from ALBERT - No Answer

11. Next looking at the TPR and TNR values for all of these results we see roughly a $\%15$ percent increase when compared to their Model 1 counter parts. The highest values of which again are in the RoBERTa - Blank from **Figure 11**.

Next two things we want to focus on was the difference between the RoBERTa and ALBERT models and the "Blank" and "No Answer" datasets.

Focusing first on the difference between RoBERTa and ALBERT, we see a much clearer difference in performance here there being an overall accuracy, TPR and TNR increase of about $\%10$. This difference is probably due to the nature of RoBERTa being a much more robust and complex model than ALBERT and is able to capture more information and subtlety in the data to thus more accurately predict whether a question is answerable or not.

Lastly, in regards to looking at the difference between the "Blank" and "No Answer" datasets, we see that we still run into the same pattern that we found with Model 1. Where the "Blank" trained models tend to obtain slightly better True Positive rates, but the models trained on the "No Answer" datasets

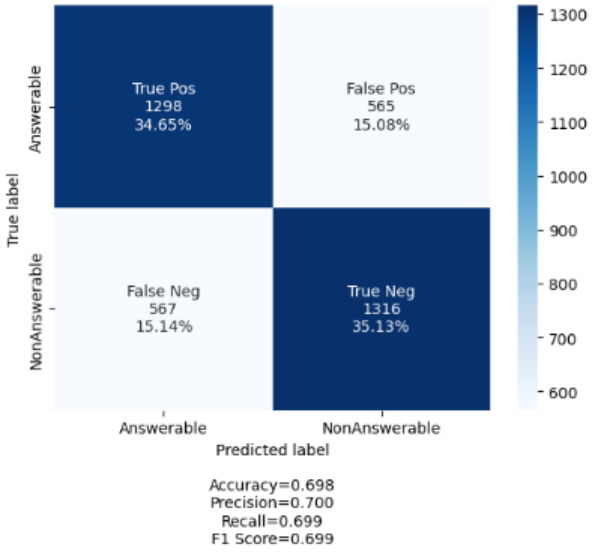

Figure 11: Confusion Matrix for Model 2 from RoBERTa - Blank

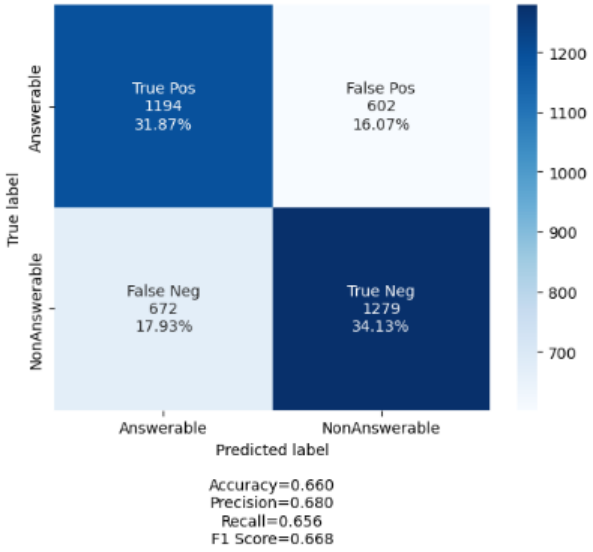

Figure 12: Confusion Matrix for Model 2 from RoBERTa - No Answer

do slightly better with the True Negative rates. This time around the difference wasn't as large as with Model 1 (more around $\%5$ percent difference. But it was still curious that this happens.

7 Final Table

Table 2 shows the accuracy on answerable and non-answerable questions for each model tested in this study to provide everything tested in one nice table to visualize. Here the first letter represents either Base (B), Model 1 (M1) or Model 2 (M2) along with the has answer accuracy (HA) and non-answerable accuracy (NA).

Table 2: Final Results

Model + Dataset	B - HA	B - NA	M1 - HA	M1 - NA	M2 - HA	M2 - NA
ALBERT - Blank	89.48	0.00	0.4486	0.5340	0.6131	0.6048
ALBERT - No Answer	89.42	0.12	0.5520	0.5441	0.5659	0.5654
RoBERTA - Blank	91.27	0.00	0.5426	0.4918	0.6399	0.6800
RoBERTA - No Answer	91.63	0.07	0.6495	0.3004	0.6960	0.6996

7.1 Overall Discussion

Tying the results from both the Model 1 training and Model 2 training, it appears that there are 3 trends that appear. The first being that models that are trained on the "Blank" dataset tend to have higher TPR values and datasets trained on the "No Answer" dataset tend to have higher TNR values. The second being that the reader model used did seem to have more of an impact on Model 2 than it did on Model 1. Lastly, we see that, overall, Model 2 was able to increase overall accuracy, TPR and TNR values by at least 10%. While this value is slightly low, and still not near the original accuracies shown in the original paper in Hu et al. (2018) we are still happy with the results.

One of the explainations that we can offer as to why Model 2 was better than Model 1 is mainly due to the fact that it provides special embedding and attention to both the sentence answer and the question when training through the entire model as opposed to Model 1 only concatinating everything and running it through a transformer. Due to the extra information and learning that Model 2 provides it would be able to learn the nuances between the sentence answer and the question to better predict a questions answerability.

8 Future Work

Future works could work on improving the performance of the models. It is clear that simply replacing the LSTMs in the second verifier model with transformers does not result in good performance.

Additionally, it is clear the verifiers are much better than the reader models at identifying unanswerable questions, but that they have a false negative rate that is much too high (too many answerable questions are marked as unanswerable). One avenue for further research involves identifying what aspects of the input are resulting in these false negatives by training on an erasure dataset; the attached codebase contains a hand-coded erasure dataset that can be used for this purpose.

Attention maps could also be used to identify how the model is determining non-answerability. Additionally, ablation studies could be performed to identify if any part of the architecture is harming performance, or could be reversed to see if adding more elements (i.e. more linear layers) could help the model learn more helpful associations.

9 Conclusion

While we were not able to reproduce the results from the paper Hu et al. (2018) we we were able to get almost comparable results with our Model 2 implementation by getting an overall accuracy of 0.70 with true positive and true negative rates of 0.696 and 0.699 as shown in **Table 2**. We saw that comparing Model 1 and Model 2 we were able to get an overall increase of accuracies with Model 2 and a significant increase in the non-answerable questions accuracy. While we were not able to reproduce the results that the original paper got, we did change their Model 2 architecture while trying 2 different reader models to see if more updated reader models would provide a higher overall accuracy.

10 Repository

All code and visualizations that were used through this report are in the following github repository:

HERE[1]

[1]Link: (https://github.com/cLawson101/CS-388-NLP-Final-Project) in case hyperlink does not work

The README file will contain files of interest so as to be able to navigate through the files.

References

Giffari Alfarizy and Rila Mandala. 2022. Verification of Unanswerable Questions in the Question Answering System using Sentence-BERT and Cosine Similarity. In *2022 9th International Conference on Advanced Informatics: Concepts, Theory and Applications (ICAICTA)*. 1–6. https://doi.org/10.1109/ICAICTA56449.2022.9932903

Minghao Hu, Yuxing Peng, and Xipeng Qiu. 2017. Mnemonic Reader for Machine Comprehension. *CoRR* abs/1705.02798 (2017). arXiv:1705.02798 http://arxiv.org/abs/1705.02798

Minghao Hu, Furu Wei, Yuxing Peng, Zhen Huang, Nan Yang, and Dongsheng Li. 2018. Read + Verify: Machine Reading Comprehension with Unanswerable Questions. https://doi.org/10.48550/ARXIV.1808.05759

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *CoRR* abs/1909.11942 (2019). arXiv:1909.11942 http://arxiv.org/abs/1909.11942

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR* abs/1907.11692 (2019). arXiv:1907.11692 http://arxiv.org/abs/1907.11692

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know What You Don't Know: Unanswerable Questions for SQuAD. *CoRR* abs/1806.03822 (2018). arXiv:1806.03822 http://arxiv.org/abs/1806.03822