Attending in BiDAF and QANet

Stanford CS224N Default Project: IID track
Mentor: Gaurab Banerjee

Lucas Orts
Department of Computer Science
Stanford University
leorts@stanford.edu

Yanal Qushair
Department of Computer Science
Stanford University
yqushair@stanford.edu

Sophia Sanchez
Department of Computer Science
Stanford University
sanchezs@stanford.edu

Abstract

In this paper, we build a question answering (QA) system and apply it on the Stanford Question Answering Dataset (SQuAD) version 2.0. Our goal is to achieve strong performance on this task without using pre-trained language models by exploring two of the most widely used architectures for this task: RNN-based and Transformer-based models. Our primary contribution are two high-performing models: a BiDAF model with character-embeddding, self- and co-attention and a QANet model. We analyze their performance and also provide some qualitative analysis of our different model's prdicted answers. Using our BiDAF model with self- and co-attention, we achieve an F1 score of 63.044 and an EM score of 59.476 on the unseen SQUAD 2.0 test set.

1 Introduction

The Question Answering Task (QA) is a salient problem in the field of NLP. Studying models designed for the QA task can improve the services of QA systems currently employed in the world today, such as conversational home assistants like Alexa and Google Home, information retrieval for user-facing interfaces, and automated text reading comprehension. Further, QA model improvement can serve as a vehicle for innovation in general NLP deep learning since good performance on the QA task requires a comprehensive, semantic understanding of natural language beyond pattern matching and statistical exploitation. Our project explores some of the approaches used by state-of-the-art QA systems by implementing them (some per the original specifications, others with modifications), testing their performance on the popular standardized SQuAD 2.0 dataset [1], and analyzing when and why the models fail. Specifically, we explore the performance of Bidirectional Attention Flow (BiDAF), which employs context-query attention, dynamic co-attention (DCN), which uses a second-level attention computation, self-attention which attends the value vectors to themselves, and QANet, which leverages self-attention, doing away with RNNs altogether. We analyze both the quantitative and qualitative performance of all these attention mechanisms.

2 Related Work

In order to infer an answer, QA systems need to compute interactions between the question and the context. This is exactly the idea behind Bi-Directional Attention Flow (BiDAF) [2], which introduces a bi-directional attention which flows not only from the question to the context but also from the context to the question. This allows the model to build question-aware representations of the context, which are then transformed into a span prediction using a long short-term memory (LSTM) network.

When first introduced, BiDAF achieved a high position on the SQuAD leaderboard, and many modifications to the model have since then increased the performance, including the Dynamic co-attention network [3], which added a second-level attention to the first-level context-query attention used by BiDAF, and R-Net [4] which adds a self-attention layer called Self-Matching Attention where the query vector is from the same set as the set of value vectors. In this paper, we explore both the idea of co-attention and self-attention.

Then, once the Transformer popularized the use of deep attention-based networks, the QANet model [5] proposed a model for QA using only self-attention and convolutions, replacing sequence-aligned RNNs entirely. While QANet retains the same context-query attention to translate meaning between the context and question as in BiDAF, it replaces the recurrence-based encoder with a transformer-based one, which empirically resulted in not only better performance but also faster training time.

3 Approach

In this project, we worked on two main models. We extended the given BiDAF baseline model by adding character embedding and modifying the attention layer, and we implemented QANet from scratch, with slight modifications from the original paper.

3.1 Bi-Directional Attention Flow (BiDAF)

The default starter-code provides us with a baseline implementation of the Bi-Directional Attention Flo (BiDAF) model [2]. This model consists of five layers: (1) embedding layer, (2) encoder layer, (3) attention layer, (3) modeling layer, and (5) output layer.

Character Embedding. As an initial extention to the BiDAF model, we implemented character-level embeddings to condition on the internal structure of words and better handle out-of-vocabulary words in the first embedding layer. For each word, we concatenate an additional character-level embedding onto the GloVe vector using an embedding lookup that converts the character indices. Specifically, for word i with GloVe vector g_i and character-level embedding c_i, we define its embedding as $e_i = [g_i; c_i]$. The latter is passed to a two-layer Highway Network, whose outputs are two sequences of d-dimensional vectors for the context and query respectively.

Co-Attention Layer. The original BiDAF context-query attention layer consists of outputs a_i, b_j defined by

$$a_i = \sum_{j=1}^{M+1} \alpha_j^i q'_j \qquad \text{and} \qquad b_j = \sum_{i=1}^{N+1} \beta_j^i c_i,$$

where the weights $\alpha^i = \text{softmax}(L_{i,:}) \in R^{M+1}$ and $\beta^j = \text{softmax}(L_{:,j}) \in \mathbb{R}^{N+1}$ corresponding to the softmax distribution over columns and rows of the affinity matrix L, are multiplied with the question and context representations, q' and c, respectively. Inspired by the Dynamic Coattention Network [3], we added a second-level attention calculation to compute the attention between the context and question representations. Specifically, we combine the α_j weights with the b_j outputs to compute the second-level attention as in [3]:

$$s_i = \sum_{j=1}^{M+1} \alpha_j b_j \in \mathbb{R}^\ell$$

Self-Attention Layer. BiDAF's attention mechanism creates and uses question-aware context representations to capture meaningful relationships between the context and question. However, BiDAF's attention has limited knowledge of the context itself, especially between respective parts of the context. In our project, we implement a self-attention layer, which directly matches the question-aware passage representation against itself to inform the model where it should attend. Inspired by R-Net [4], our model's self-attention layer complements our model's Context-to-Question attention layer. However, we implement the self-attention differently from R-Net's implementation—our self-attention implementation generates a similarity matrix where the query vector is from the same set as the set of value vectors. Specifically, suppose we have context hidden states $c_1, ..., c_N \in \mathbb{R}^{2H}$, where H is the hidden size. We compute the similarity matrix $S \in \mathbb{R}^{N \times N}$, such that S_{ij} represents

the similarity between the context hidden states c_i and c_j.

$$S_{ij} = w_{\text{sim}}^T[c_i; c_j; c_i \circ c_j] \in \mathbb{R}$$

where $w_{\text{sim}} \in \mathbb{R}^{6H}$ is a weight vector. Next, we perform self-attention on the context representation by taking the row-wise softmax of S. We use that result to take weighted sums of the context hidden states c_i to get context-to-context attention output a_i. Mathematically,

$$\bar{S}_{i,:} = \text{softmax}(S_{i,:}) \in \mathbb{R}^N \quad \forall i \in 1, ..., N$$

$$a_i = \sum_{j=1}^{N} \bar{S}_{i,:} c_j$$

Lastly, for each context location $i \in \{1, ..., N\}$, we obtain the output g_i of the self-attention layer by combining the context hidden state c_i and the self-attention output a_i:

$$g_i = [a_i; c_i \circ a_i],$$

which is then concatenated onto whatever bidirectional attention flow the model employs.

3.2 QANet

QANet [5] adapts ideas from the Transformer architecture and applies them to question answering, replacing RNNs entirely with self-attention and convolution. For our project, we implement QANet from scratch, reusing certain implementations of the BiDAF baseline model. The architecture of QANet is similar to that of the BiDAF model, with the same five layers, as shown in Figure 1.

Input Embedding Layer. The QANet embedding layer consists of both word and character-level embeddings. We used our implementation of character-level embeddings from BiDAF.

Embedding Encoder Layer. Instead of using an LSTM-based encoder, the QANet's embedding encoder layer consists of a positional-encoding, consisting of sinusoidal functions at varying wavelengths, and three residual sub-blocks: convolutional (consisting of depthwise separable convolutions), self-attention (with a multi-headed attention mechanism) and feed-forward net.

Context-Query Attention Layer and Model Encoder Layer. For the attention layer, we reused the BiDAF baseline model's attention layer, whose output we pass through a series of three stacked model encoder blocks in the model encoder layer. The model encoder block has the same structure as the embedding encoder block, with a different number of convolution and model blocks.

Output Layer. As in the BiDAF model, QANet has a final output layer, which calculates the probabilities of the start and end answer pointer, modeled as

$$p^1 = \text{softmax}(W_1[M_0; M_1]), \quad p^2 = \text{softmax}(W_2[M_0; M_2])$$

respectively where M_0, M_1 and M_2 are the outputs of the model encoder layer (from bottom to top) and W_1, W_2 are trainable variables. The loss function is the same as for the BiDAF model.

4 Experiments

4.1 Data

Our model was trained and evaluated on data sourced from the Stanford Question Answering Dataset (SQuAD) 2.0 [1]. This dataset is composed of a set of $\langle$`question, context, answer`$\rangle$ triples. The `context` is a excerpt of text from Wikipedia. The `question` is the question that the model must attempt to answer based on the content of the `context`. Notably, not all the questions in SQuAD 2.0 can be answered based on the `context`. Finally, the `answer` is a human-provided span of text from the `context` that correctly answers the `question`, if such an answer exists in the `context`. As such, the `answer` (if it exists) is necessarily a substring of the `context`, meaning that the model must identify the `answer` within the text of the `context` as opposed to generating an answer from scratch.

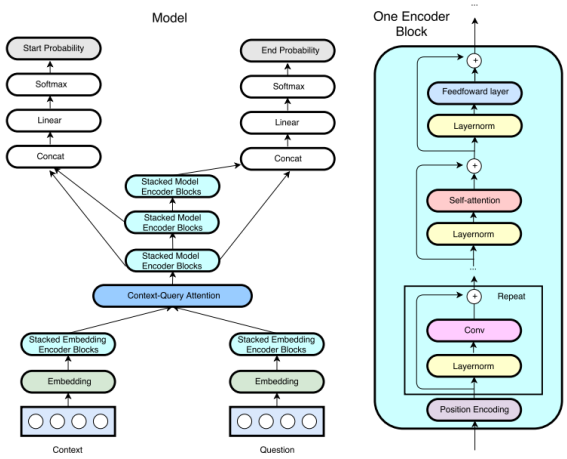

Figure 1: The QANet Architecture Diagram.

We trained our model on the 129,941 examples that comprise the original SQuAD 2.0 training set. Our dev set and test set are respectively 6,078 examples and 5915 examples drawn from the original SQuAD 2.0 dev set. Drawing both our dev and test sets from the dev set is necessary as the original test set is kept secret for class leaderboard purposes. Note that each training example contains a single answer, while each dev and test example contains three answers for evaluation purposes, as described below.

4.2 Evaluation method

The model's performance is evaluated using two metrics: Exact Match score (EM) and F1 score.

Exact Match. The EM score on a single evaluation (dev or test) example is a binary score indicating whether the model's predicted output *exactly* matches at least one of the three human-provided answers.

F1 score. The F1 score is the harmonic mean of the model's precision and recall on a given evaluation example. Precision is the percentage of words in the model's returned answer that can be found in the ground-truth answer, while recall is the percentage of words in the ground-truth answer that are included in the model's answer.

For each evaluation example, EM and F1 scores are calculated between the model's output and each of the three human-provided answers. Then, the maximum EM score and maximum F1 score are stored for that example. Finally, the scores are average over all evaluation examples to get a final F1 and EM score for the model.

4.3 Experimental details

Report how you ran your experiments (e.g. model configurations, learning rate, training time, etc.)

4.4 Results

	EM	F1
Baseline	58.01	61.25
Character-Embedding (CE)	60.39	63.71
CE, Self- and Co-attention	61.25	64.65
QANet (hidden 128, att. heads 1)	53.77	57.23
QANet (hidden 64, att. heads 8)	61.27	64.32

Table 1: Dev. Set EM and F1 Scores of notable models.

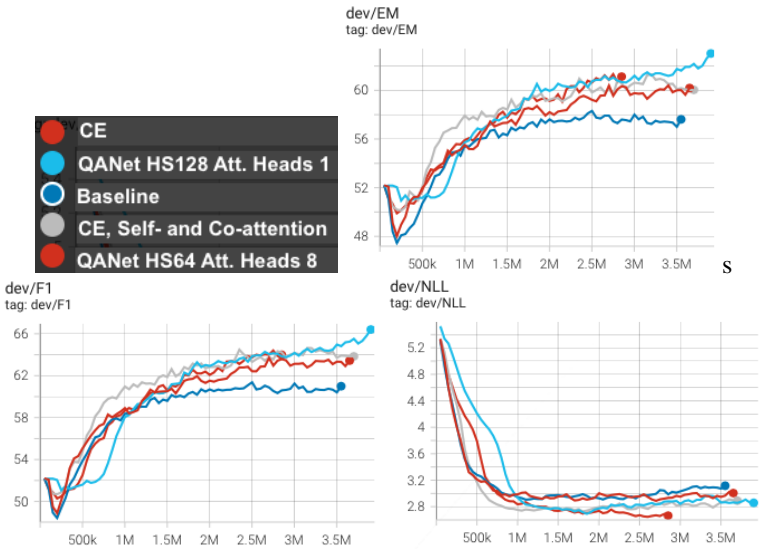

Figure 2: F1, EM and NLL TensorBoard plots for each model.

4.4.1 BiDAF Baseline

Baseline Model. We used the default hyperparameters, which were hidden size: 100, learning rate: 0.5, dropout rate: 0.2, batch size: 64, ema decay: 0.999. For most of our BiDAF based models, we did not modify these hyperparameters except for hidden size and batch size.

Character Embedding. We did not modify the hyperparameters from the baseline.

Coattention. All of the hyperparameters remained constant except for hidden size. We tried models respectively with hidden sizes of 100 and 200. Both models achieved scores significantly below the baseline BiDAF model, with F1 scores of ∼58 and EM scores of ∼52.

Self-attention. We attempted a wide range of different hidden sizes and both the BiDAF context-query attention and coattention as the bidirectional attention base for our self-attention module. We found that a model that used character embedding, self-attention, and coattention, with hidden size of 100, performed well. Additionally, we found this configuration with a larger hidden size increased training time without significantly improving performance.

Of the BiDAF models listed above, the BiDAF model that used character embedding, self-attention, and coattention achieved the best performance with a EM of 61.25 and F1 of 64.65. Our group is surprised that the models with only coattention performed so poorly, while the model with both coattention and self-attention is one of our group's best performing models. Most likely, the coattention only models could achieve better perform after further hyperparameter tuning.

4.4.2 QANet

QANet with single-headed attention. For this QANet model, we used the following parameters: hidden size: 128, batch size: 32, number of attention heads: 1, learning rate: 0.5, dropout rate: 0.2, number of blocks in the embedding encoder: 7, embedding encoding kernel size: 7, number of convolutions in embedding encoder: 4, number of blocks in the model encoder: 7, model encoder kernel size: 5, number of convolutions in model encoder: 2. These parameters are consistent with the parameters listed by the original QANet paper besides the attention head count, the learning rate, and dropout rate. We were unable to increase the number of attention heads of this model due to the memory constraints of our GPU hardware.

QANet with multi-headed attention. This iteration of the QANet model had the same parameters as the previous model, except for the following: hidden size: 64, number of attention heads: 8.

As seen in table 1, the single-headed QANet model achieved poor EM and F1 scores of 53.77 and 57.23, respectively while the multi-headed QANet model achieved comparatively better EM and

F1 scores of 61.27 and 64.32. The single-headed QANet model's poor test performance but best training performance, as seen in plots 2, indicate that the model overfit to the dev set. Meanwhile, the multi-headed QANet model achieved a performance comparable to our group's best BiDAF model. In the QANet paper, QANet was shown to be empirically better in EM and F1 scores than BiDAF, yet our best QANet model is only about equal to our best BiDAF based model. One difference between our implementation from the original QANet implementation is that for all of our feed-forward layers, we used a linear layer. In contrast, the original QANet implementation employed 1-D convolutions for their feed-forward layers. Due to time constraints, we were unable to test if this implementation difference is the source of the performance discrepancy. Additionally, due to the long training times of the QANet model, we did not perform a large parameter sweep to better tune our QANet model, which could have potentially improved our models' performance significantly.

5 Analysis

In this section, we examine the different models to see how they work, when they succeed and when they fail.

5.1 Evaluation by Question Type

We split the entire dev set into questions starting with "Who", "When", "Which", "What", "Where", "How" and "Why" and evaluated the F1 and EM scores for each model. We plot the results in Figure 3. The results suggest the natural intuition that more abstract questions, i.e. questions that start with

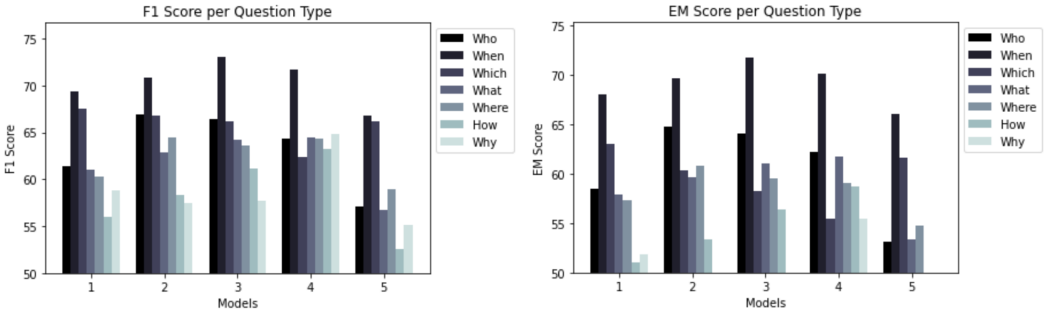

Figure 3: F1 and EM Scores per question type. Model 1 is the BiDAF baseline, 2 is the CE model, 3 is the CE, self- and coattention model, 4 is the multi-headed QANet model, and 5 is the single-headed QANet model. 1

the words "How" or "Why", are harder for the model to correctly predict the answer for. Even with the simpler questions such as those starting with "Which", all models seem to struggle to find the *exact* match. Notably, the F1 scores achieved for this category are much better than the EM score, highlighting the importance of the dual evaluation metric. Finally, observe that the best QANet model (4) achieves a significantly better F1 score for "Why" and, to a lesser extent, "How" than any of the other models. This suggests that the QANet architecture is more capable of better understanding more abstract relationships within contexts than the BiDAF based model.

5.2 Evaluation by Context Length

Similarly to the previous subsection, we split the entire dev set into questions with context lengths of increasing size. We plot the F1 and EM score for each context length bucket in Figure 4.

The most noticeable result is that our best BiDAF model, which utilizes both co-attention and self-attention, performs significantly better than any other model on very small contexts, i.e. between 0 and 200 context tokens. This is probably due to the fact that when the context is very small, the different types of attention have less to attend to and it is probably easier to understand what is most important within that context window. Now, for QANet there seems to be a very slight tendency to do better as the context window grows, but those improvements are marginal once the context length

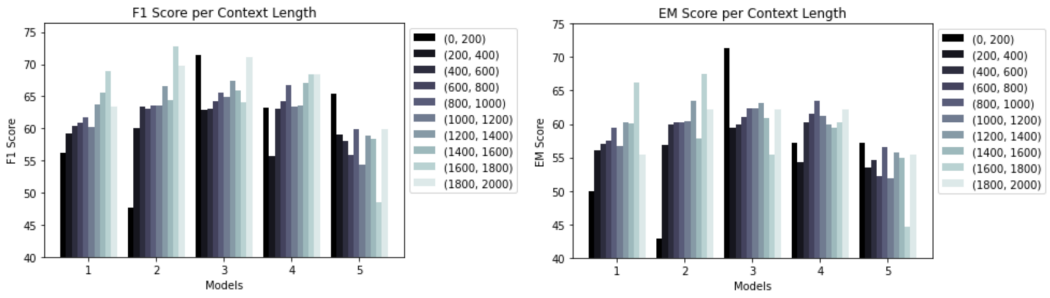

Figure 4: F1 and EM Scores per context length. Model 1 is the BiDAF baseline, 2 is the CE model, 3 is the CE, self- and coattention model, 4 is the multi-headed QANet model, and 5 is the single-headed QANet model. 1

passes 400. Thus, this suggest that past a certain minimal context length, QANet is able to understand the context well enough to parse it and perform the QA task.

5.3 Distribution of Error Types

We also consider how the model fails. Specifically, for each model we looked at its error and categorized it into three categories: (1) NA False Positive (FP), which is when the model returns NA when a correct answer is in the context, (2) NA False Negative (FN), which is when the model returns an answer when there is no correct answer within the context, and (3) No Match, which is when the model returns an answer that does not match one of the correct answers. We plot the results in Figure 5.

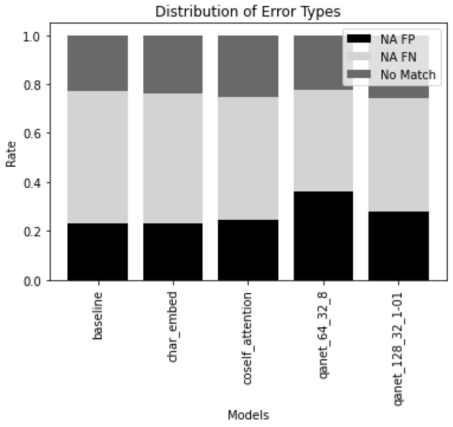

Figure 5: False positive, False Negative, and No Match Error Rates for our 5 notable models

From Figure 5, we see that for all models, almost 80% of their errors are related to not being able to identify whether the answer is in the given context or not. This is understandable since identifying whether the answer is in the context demands a level of semantic understanding of the context, not just simple pattern matching. Now, for almost all the models, the FP and FN rates significantly dominate the no match rate, which means that the models are somehow incentivized to return an answer. We thought this may relate to the underlying number of questions with context-contained answers and number without context-contained answers being skewed, but these two numbers turned out to be virtually evenly distributed. Only model (4) has a relatively even split between NA FP and NA FN, which again suggests that QANet is predisposed to have a better understanding of the contexts than our BiDAF model implementations. In Figure 7, we have a specific output of when the QANet models correctly classifies a question as NA, while even our best BiDAF model fails to do so.

6 Conclusion

In this paper, we have implemented and explored different attention-based models for the SQuAD 2.0 question-answering task. Our two highest performing models were our BiDAF model with character-embedding, self- and co-attention and our QANet model with hidden size of 64 and 8 attention head which achieved EM scores of 61.25 and 61.27 and F1 scores of 64.65 and 64.32, respectively. We submitted the former model to the Test Leaderboard and achieved an EM score of 59.476 and a F1 score of 63.044.

For each of our implemented models, we analyzed their quantitative and qualitative performance. Though our implementations achieved improvements over the baseline, they still underperformed the state-of-the-art implementations from which they were based on. Here we take some time to reflect on some limitations of our work and directions for future work. The limited hyperparameter tuning we did for our co-attention models and for QANet showed very varied performance, suggesting that hyperparameter tuning is salient for achieving high performance. However, due to both time and hardware constraints we we did not do extensive hyperparameter tuning and we also had to limit our models' size. To achieve a singular model which combines the respective strengths of BiDAF and QANet, specifically for the different question types, our group thought that ensembling the models together may be promising. Finally, even though we restricted ourselves from using pre-trained models (i.e. we couldn't use models such as ALBERT or ELECTRA), we could still have improved each individual models with other extensions, leveraging techniques such as data augmentation or active learning. Additionally, further examining our attention mechanism through attention heat maps would likely provide us with valuable insight into why our individual models are or are not performing as expected.

Finally, with our implementations and analysis of BiDAF and QANet—an RNN and Transformed-based model respectively—we have seen how both architectures struggle in capturing long-term dependencies. RNNs, in particular Long Short Term Memory (LSTM) networks, can in theory have unlimited context windows but in practice are still limited in learning long-term dependencies due to the vanishing gradient problem. Meanwhile, Transformers have fixed-length context windows such that the maximal dependency distance between two words is limited to the training input's length. With more time, we would have finished our implementation of the *Transformer-XL* model, which attempts to address this shared shortcoming by enabling learning dependencies beyond the fixed length context.

References

[1] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Association for Computational Linguistics (ACL)*, 2018.

[2] Ali Farhadi Minjoon Seo, Aniruddha Kembhavi and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

[3] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. *arXiv:1611.01604*, 2018.

[4] Microsoft Research Asia Natural Language Computing Group. R-net: Machine reading comprehension with self-matching networks. 2017.

[5] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*, 2018.

A Appendix

Question. What persons were not allowed to settle in New France?
Context. The exodus of Huguenots from France created a brain drain, as many Huguenots had occupied important places in society. The kingdom did not fully recover for years. The French crown's refusal to allow non-Catholics to settle in New France may help to explain that colony's slow rate of population growth compared to that of the neighbouring British colonies, which opened settlement to religious dissenters. By the time of the French and Indian War (the North American front of the Seven Years' War), a sizeable population of Huguenot descent lived in the British colonies, and many participated in the British defeat of New France in 1759-60.
Answer. non-Catholics

Figure 6: Model [3] Predicted Answer: non-Catholics. Model [4] Predicted Answer: French crown's refusal. This is an example of when [3] correctly predicted the answer but [4] did not.

Question. What kind of force did Harthacnut establish?
Context. When finally Edward the Confessor returned from his father's refuge in 1041, at the invitation of his half-brother Harthacnut, he brought with him a Norman-educated mind. He also brought many Norman counsellors and fighters, some of whom established an English cavalry force. This concept never really took root, but it is a typical example of the attitudes of Edward. He appointed Robert of Jumièges archbishop of Canterbury and made Ralph the Timid earl of Hereford. He invited his brother-in-law Eustace II, Count of Boulogne to his court in 1051, an event which resulted in the greatest of early conflicts between Saxon and Norman and ultimately resulted in the exile of Earl Godwin of Wessex.
Answer. *No Answer*.

Figure 7: Model [3] Predicted Answer: *No Answer*. Model [4] Predicted Answer: English cavalry force. This is an example of when [4] correctly predicted the answer but [3] did not.