A Deep Learning Approach for Identification of Confusion in Unstructured Crowdsourced Annotations

Rachel Gardner
Stanford University
rachel0@stanford.edu

Maya Varma
Stanford University
mvarma2@stanford.edu

Clare Zhu
Stanford University
clarezhu@stanford.edu

Abstract

Large, densely-labeled datasets have catalyzed the creation of effective deep learning methods for text and image analysis, leveraging the power of natural language data found in massive online forums such as Reddit. While this form of social media-based crowdsourced data comes at a lower cost than hiring annotators, it is also more prone to the pitfalls of natural language answering, since data is unstructured and respondents might not answer the provided questions correctly. To address these issues, we present a deep learning approach to systematically identify confusion and extract answers from unstructured crowdsourced annotations of data collected from Instagram. Each annotation contains an image, a machine-generated question, and a crowd-sourced response: we use a top-down bottom-up visual attention model based on Facebook AI Research's Pythia architecture to 1) classify whether the response is an example of confusion, and 2) extract the ground-truth answer from the natural-language response. We repeat these tasks with a BERT-based classifier, using text features from the question and the response (excluding image features). Our results show that surprisingly, the BERT-based model (classification AUC-ROC=0.84, answer prediction F1=0.77) outperforms the modified Pythia architecture (classification AUC-ROC=0.79, answer prediction F1=0.46) , despite the lack of image features. Furthermore, a multi-task BERT-based model trained simultaneously on tasks (1) and (2) is able to outperform task-specific models (classification AUC-ROC=0.84, answer prediction F1=0.78). The method presented in this work holds potential for reducing the need for manual quality analysis of crowdsourced data as well as enabling the use of annotations from unstructured environments such as social media platforms.

1 Introduction

The recent revolution in the development of deep learning methods for automated analysis of text and images has mandated the creation of large, densely-annotated training datasets, which are often labeled by paid workers on Amazon Mechanical Turk. Although obtaining dataset annotations through crowdsourcing enables the creation of gold standard labels that have been vetted by humans, it is a time-consuming and expensive process; for example, the Visual Genome dataset, which includes dense annotations on over 100k images, involved more than 33,000 paid workers over a six-month time period (13). As the demand for large, labeled datasets continues to increase, there is a pressing need for accurate and efficient evaluation of crowdsourced annotations. In this project, we explore the use of deep learning and natural language processing techniques to automatically identify user confusion in unstructured crowdsourced data labels. Our dataset contains images, computer-generated questions referring to each image, and responses from social media users. We determine that a user response shows confusion if (1) the response contains an incorrect or irrelevant answer or (2) the question is impossible to answer from the given image.

A preliminary analysis of the data showed that human evaluators can often accurately identify confusion in responses without consulting the corresponding image [1]. Thus, to investigate the role of image features in classifier performance, we divide our confusion detection problem into two subtasks: (1) Visual Question Response (VQR), which involves the analysis of questions, user responses, and image features, and (2) Question Response (QR), which relies solely on questions and responses. For each task, we first perform a binary classification of confusion by predicting a binary label (with 0 representing the presence of a correct answer and 1 representing confusion as defined above); then, we identify the correct answer from the unstructured response text.

The creation of automated methods for evaluating accuracy of crowdsourced labels will greatly improve the modern deep learning workflow paradigm by reducing the need for manual quality analysis of crowdsourced data as well as enabling the use of annotations from users on social media platforms.

2 Related Work

To the best of our knowledge, no previous studies have attempted to utilize deep learning approaches for identifying confusion in unstructured user responses. In this section, we discuss related approaches that involve the use of image features and text embeddings for classifying data.

2.1 Visual Question Response (VQR) Task

A 2017 study conducted by Microsoft Research proposes a combined bottom-up and top-down attention mechanism to extract features from images: (1) a Faster R-CNN model is used to identify salient features (bottom-up), and (2) the question text is used as context to weight these features (top-down) (1). Features from the question text are extracted and combined with the image features, generating a joint embedding of the image and question.

Pythia, a model designed by Facebook AI Research, was an entry to the 2018 Visual Question Answering (VQA) Challenge, where it achieved the top performance (accuracy 72.27%) on the vqa_v2.0 dataset (18; 12; 2; 19; 8). Pythia uses the bottom-up and top-down attention model as a baseline, making a series of key modifications, such as changes to attention mechanisms and the utilization of detectors based on Feature Pyramid Networks from Detectron to extract features. (7).

Since the Pythia model presents an effective method of combining features across images and text data, we found it to be a suitable starting point for our VQR task; however, significant modifications were necessary due to the nature of our task, as detailed in subsection 4.2.

2.2 Question Response (QR) Task

Pretrained contextual word representations have been shown to improve machine understanding of language (16; 10). A 2018 study conducted at Google AI proposes a novel transfer learning method called BERT (Bidirectional Encoder Representations for Transformers) for generating contextual encodings of words (5). The BERT approach is an unsupervised learning method that involves training a deep bidirectional language model with transformers and then using the learned encodings in other NLP tasks.

The QR task relies on text-based features from the question and user response, so we found the BERT method to be an appropriate starting point. Since the QR task differs from standard NLP tasks like question-answer prediction, customization was necessary. Our modifications to the BERT architecture are outlined in subsection 4.3.

3 Data

The data for this project includes 50,628 Image-Question-Response trios, which were obtained from social media. Users who uploaded public photos on Instagram were asked questions by a bot based

[1] Respondents often specifically point out why the question asked was incorrect. In such cases, human evaluators can identify confusion without examining the image.

on the features of the images, and the user responses were collected for inclusion in this dataset [2]. User responses come in the form of unstructured natural language data with colloquial language, spelling errors, and emojis; the mean length of a response is 35.9 characters or 6.8 words. All trios in our dataset were manually annotated with the ground truth answer by Amazon Turk workers.

We first assigned binary labels to indicate the presence of confusion in user responses, assigning a label of 1 if the Amazon Turk annotator could not identify a correct answer in the user response, and assigning a label of 0 otherwise.

We then added additional annotations to all examples with accurate user responses (label=0) in order to facilitate identification of the correct answer phrase in the unstructured responses. We designed a custom, emoji-aware tokenizer to extract tokens from responses; then, we annotated all examples with the answer span, consisting of the starting and ending index of the ground truth answer with respect to the response tokenization. Since the ground truth answers in our dataset were manually composed by human annotators, there is considerable noise; in many cases, the ground truth answers are not an exact match or substring of the user response. Thus, span identification was performed with a fuzzy string matching algorithm. Significant development effort went into designing tokenization and span extraction methods to account for a variety of edge cases (which are extremely common in unstructured responses)In some cases, the ground truth answer could not be found in the user response despite having been labelled as not confused (label=0) in the previous step; these examples were removed from our dataset. [3]

We randomly split our dataset into train (80%), validation (10%), and held-out test sets (10%), The dataset shows substantial class imbalance, with nearly twice as many examples assigned labels of 0 as those assigned labels of 1. Table 4 (in the supplementary material) shows the distribution of our dataset.

4 Approach

4.1 Baseline Models

We devised two baseline models to identify confusion in user responses to questions. Both baselines make binary predictions by evaluating text-based features from questions and responses, without considering image features.

For our initial baseline approach, we designed a bag-of-words model. Naïve tokenization of questions and responses was performed based on whitespace characters, and a fixed vocabulary was generated from the 10,000 most frequent words appearing in the training set. All questions and responses were then encoded as bag-of-words frequency vectors of size 10,000. The question vector and response vector were individually passed through separate fully-connected layers and ReLU nonlinearities, before being concatenated and passed through a final fully-connected layer and sigmoid nonlinearity. This resulted in a single value representing a probability of confusion.

Our second baseline model involved the use of a more complex neural model as well as a different method for encoding the input. Again, naïve tokenization was performed. Each token in the question and user response was represented with either a 300-dimensional GloVe vector or a 300-dimensional emoji2vec representation (15; 6). Then, the encoded question and user response were passed through separate single-layer, unidirectional LSTMs (9). The final hidden states of the LSTMs were concatenated and passed through a fully-connected layer and a dropout layer (p=0.5); finally, we applied a sigmoid nonlinearity to the output to obtain a probability of confusion.

4.2 Visual Question Response (VQR)

The Pythia model served as an effective starting point, but major changes to the architecture were necessary to perform the VQR task. Whereas Pythia predicts answer labels given the image and the question, our model utilizes an image-question-response trio to predict whether the response contains a correct answer to the question based on an image. Another important issue is the distinction

[2] It is important to note that since questions were asked by a bot, some questions were impossible to answer

[3] On manual inspection, these turned out to be cases where the annotator provided a summary/translation of the ground truth answer or invented an answer to the question that was not present in the original response.

between natural-language and formatted inputs. The Pythia model and standard visual question answer datasets are built to handle formatted answers only (such as "table"). The VQR dataset, on the other hand, uses natural language responses, such as "yes, it's my grandmother's table."

Our first modification to the Pythia architecture is the mapping of input data. We converted our dataset to the COCO format, adding fields for labels and responses (shown in Figure 1). We developed a custom tokenizer to handle our unstructured input, which helps address a variety of edge cases that are typically not present in formatted text data. For example, many responses contain emojis and punctuation used without separating whitespaces; our tokenizer handles these cases to avoid the unnecessary use of <UNK> word embeddings. Next, we explored two types of word embeddings to encode questions and responses: (1) a combination of 300-dimensional GLoVe and emoji2vec embeddings, and (2) 300-dimensional FastText embeddings (15; 6; 3; 14). A pretrained Faster R-CNN model from Facebook's Detectron software package was used to extract features from all images; this model computes bottom-up attention by identifying salient objects in images (11; 7).

```
entry{
    "question" : str, "what color is the ball?"
    "element_id" : int, identifying number
    "image_id": int, mapping to image
    "true answer" : str, "red"
    "label": 0 or 1, whether the answer is a substring of the response
    "response" : str, "it looks kind of red?"
}
```

Figure 1: **VQR Dataset Format** The custom format used by the VQR dataset in a single JSON entry, modeled after the Microsoft COCO dataset (4).

Next, in order to handle the user response, we added a significant extension to the Pythia architecture, as shown in Figure 2a. A word embedding of the user response is first generated. Then, the encoding is passed through an LSTM with hidden states of size 1028, followed by a dropout layer, two convolutional layers, and a final softmax layer. After obtaining this encoding of the user response, we generated a joint embedding of image, question, and response embeddings by passing each representation through separate linear layers and ReLU nonlinearities and then computing a weighted Hadamard product. In the following equation, e represents the joint embedding, f represents a linear layer, i represents the image representation, and q and r represent question and response embeddings.

$$e = (2 * f_i(i)) \circ (0.5 * f_q(q)) \circ (0.5 * f_r(r))$$

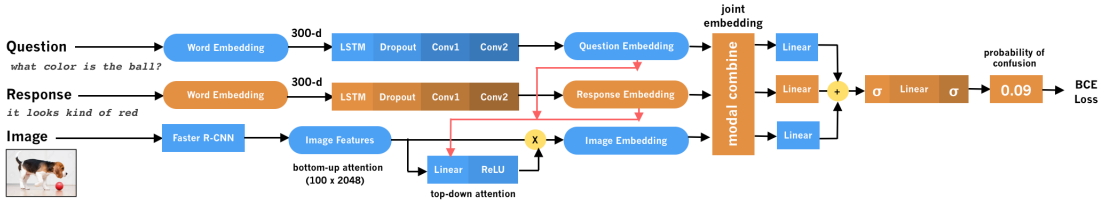
The question embedding (q) and response embedding (r) are also used to compute top-down attention on images.

For binary classification, the Pythia model was modified to produce a single class output. We passed the joint embedding through three linear layers, combined outputs, and applied a sigmoid nonlinearity to the result; this tensor was then passed through a linear layer and a second sigmoid nonlinearity. The output of the classifier is a value $0 \leq \hat{y} \leq 1$, representing the probability of user confusion. Weighted BCE loss was computed, with weightings assigned by class ratios.

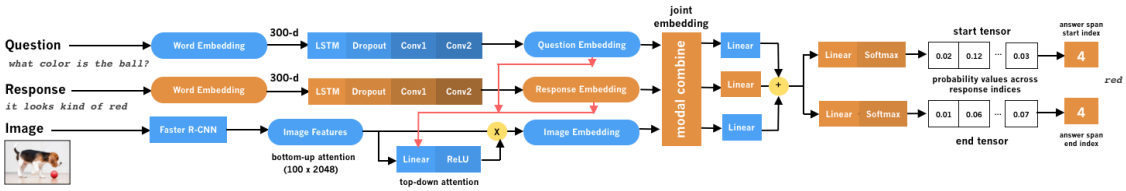
For answer prediction, the Pythia model was modified to predict the start and end index of the true answer within the user response. The joint embedding was passed through three separate nonlinear layers (each consisting of a linear layer and ReLU nonlinearity) and three linear layers. Then, outputs were combined and passed through two additional linear layers; finally, a softmax activation function was applied to the output. The two resulting tensors are of size 30, representing the maximum number of tokens in a response; the first tensor represents a probability distribution across indices likely to be the start of the answer span, and likewise, the second tensor represents a probability distribution across indices likely to be the end of the answer span. Finally, the BCE loss was computed over the predicted spans.

4.3 Question Response (QR)

The BERT architecture served as an effective starting point for the QR task; our model builds on Google AI's pretrained BERT base uncased model. For binary classification, pooled outputs from the BERT model were passed through a dropout layer and a fully connected layer, resulting in two output classes. Then, a softmax activation function was applied. The resulting value of the first output class



(a) VQR Model for Binary Classification



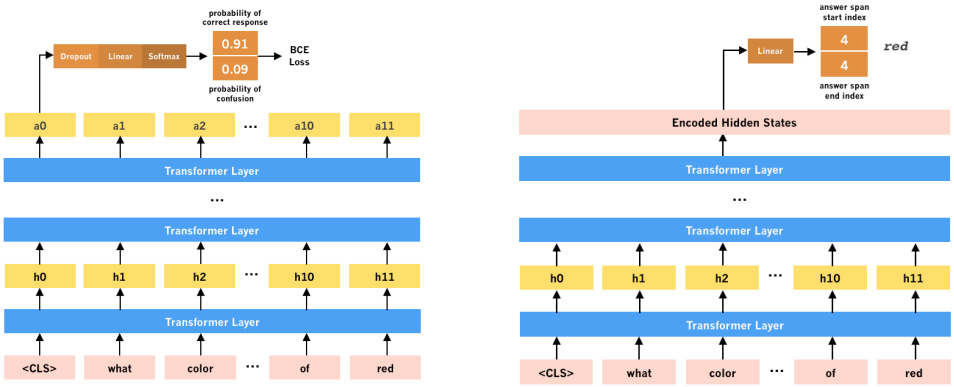
(b) VQR Model for Answer Prediction

Figure 2: **Model architecture for VQR binary classification and answer prediction** The original Pythia architecture is shown in blue, with our customizations in orange

represents the probability that the user reported a correct answer, while the value of the second class represents the likelihood of user confusion. This was done to mimic the existing structure of BERT for classification; in practice AUC-ROC is computed on the output of the second class only.

For answer prediction, encoded hidden states corresponding to the last attention block are passed through a single fully connected layer. This results in two output classes, representing indices in the response tokens; the first value represents the start index of the answer span and the second represents the end index. Since BERT performs subword tokenization, we had to perform an alignment between predicted output indices and our word-level response tokens.

We also designed a multi-task model based on the BERT architecture, which is simultaneously trained on both tasks. For each example in our dataset, the model identifies user confusion and predicts an answer span, combining loss values from both tasks; however, in the case where the user response demonstrates confusion (and consequently doesn't have an associated answer span), the loss for span extraction is manually set to zero. Thus, the answer extraction head is only updated for valid responses.



(a) QR Model for Binary Classification

(b) QR Model for Answer Prediction

Figure 3: **Model architecture for QR binary classification and answer prediction** Our modifications to the BERT architecture are in orange.

5 Experiments

5.1 Evaluation Method

We evaluated performance on the binary classification task by computing precision, recall, F1 scores, and AUC-ROC scores. Performance on the answer extraction task was evaluated with F1 and exact match (EM) scores, following the standard established by the SQuAD challenge (17). Qualitatively, we explore both correct and incorrect predictions made by the model; we also explore examples where our model produces surprising results.Since this dataset has not been previously released, our scores are the first existing performer on this dataset.

5.2 Experimental Details

For the bag of words baseline, we trained the model over three epochs with batches of size 16, utilizing the Adam optimizer with a learning rate of 0.001 and default parameters (β_1=0.9, β_2=0.99). The total training time for the model was 10 hours. For the baseline with pretrained GloVe word embeddings, we trained the model over five epochs with batches of size 32, again utilizing the Adam optimizer with a learning rate of 0.001 and default parameters (β_1=0.9, β_2=0.99). The model took approximately 18 hours to train.

For the VQR approach based on Pythia, we first preprocessed all images by using a Faster R-CNN model to extract features; this process took four days to complete. We trained the model over 12000 iterations with a batch size of 32, using the Adamax optimizer with a learning rate of 0.0001. After significantly optimizing the code base and preprocessing input data, we accomplished a 4x reduction in total training time to just 88 minutes.

For the QR approach based on BERT, we trained the model over 10 epochs with a batch size of 32, which took approximately an hour. We used the default BERT learning rate of 5e-5 for the Adam optimizer. Since BERT requires a maximum sequence length, we implemented a custom trimming method to suit our dataset. We set the combined maximum sequence length to 50; however since this represents the combined total of tokens after subword tokenization, several examples exceeded the maximum length. In these cases, our trimming method randomly chooses a window of the response such that the ground truth answer is included, discarding the rest of the response. If the user response does not contain the true answer, a random window is selected.

5.3 Results

	Binary Classification				Answer Prediction	
Model	AUC-ROC	Precision	Recall	F1	EM	F1
Baseline - Bag of Words	0.50	0.37	0.18	0.24	N/A	N/A
Baseline - GloVe Embeddings	0.74	0.62	0.46	0.53	N/A	N/A
VQR Model	0.79	0.68	0.59	0.61	0.23	0.46
QR Model	0.84	0.73	0.67	0.70	0.60	0.77
Multi-Task QR Model	0.84	0.73	0.68	0.71	0.61	0.78

Table 1: **Model Performance** Comparison of model performance on held-out test set for binary classification and answer prediction tasks

The bag-of-words baseline model achieved an AUC-ROC of 0.50 (precision=0.37, recall=0.18) on the test set. The baseline with pretrained GloVe word embeddings resulted in an AUC-ROC of 0.74 (precision=0.62, recall=0.46) on the test set. However, this baseline model achieved an AUC-ROC of only 0.774 on the training set, which indicates that the model is not powerful enough to model all of the complexities of the data. Further, both baselines took significantly longer to train than the VQR and QR models.

Interestingly, the BERT-based QR model achieved higher performance than the Pythia-based VQR model on both the binary classification and answer prediction tasks, with the Multi-Task QR model achieving an even higher performance than the single-task QR model. It is possible that including

image features diverted the VQR model's attention away from the text features of the response that were likely to represent confusion.

6 Analysis

6.1 Model Experiments

Model	AUC-ROC	Precision	Recall	F1
VQR Model - Response Attention, GloVe Embeddings	0.79	0.68	0.59	0.61
VQR Model - No Response Attention	0.78	0.71	0.52	0.58
VQR Model - FastText Embeddings	0.78	0.66	0.59	0.60

Table 2: **Binary Classification Model Experiments** Results from VQR model experimentation

Experimental results show that using text features from responses to compute attention on images led to slight improvements in model performance.

We also compared the VQR binary classification model based on GLoVe embeddings with a model based on FastText embeddings trained on Common Crawl (3; 14). Since the FastText embeddings encode subword information, we were able to generate a 300-D FastText embedding for all words in the natural-language response vocabulary, even for words that were misspelled and would normally not have a corresponding GLoVe embedding. This allowed for the removal of many <UNK>s in the dataset. Despite this improvement in encoding responses, model performance was similar to the results obtained from the VQR model trained on GLoVe embeddings. A possible explanation is that spurious relationships may exist at the character-level between word embeddings from the original Common Crawl vocabulary and word embeddings outside of the vocabulary (that were computed using subword information), [4] resulting in similar FastText embeddings for words with completely different meanings. Additionally, our data analysis revealed that many of the <UNK> tokens were concentrated in few examples, so improving those examples did little to affect the overall score; others, such as misspelled words, occurred a handful of times in a few examples, and were not always necessary for characterizing the overall confusion of a response.

6.2 Qualitative Evaluation

Analyzing individual examples is important for understanding model performance. Table 3 lists several illustrative examples for our best performing model. Note that answers are only predicted if the response is classified as correct (label = 0).

#	Question/Response	Answer	Label	Predicted Label	Predicted Answer
1.	What is on top of the cake? chilli☺ that is not cake that's chicken	N/A	1	1	N/A
2.	What is on the table? beet and carrot juice☺☺	juice	0	0	beet and carrot juice
3.	What is on the building? it's a kindergarten☺☺	N/A	1	0	kindergarten
4.	Where is the picture taken? in Vienna, Austria☺☺	N/A	1	0	Vienna, Austria
5.	Where is the dog? sat next to me on the sofa	N/A	1	0	sofa

Table 3: **Multi-Task Model Evaluation** Examples of model performance on various edge cases

[4] For example, between "false friends" of two languages: "estimated," in English, has a similar sequence of characters to "estimado" in Spanish, even though the meaning of "estimado" is closer to that of "esteemed" in English.

In example 1, the original question was unanswerable given the content of the image. Annotators were inconsistent with determining accuracy of responses to unanswerable questions; however, the model still learned to predict that the user is confused. In this case, the multi-task model predicted "chilli" as the answer, but this prediction was not reported since the binary classification assigned a label of 1. In example 2, the model correctly identifies the answer, though it is slightly more verbose than the ground truth. While annotators varied widely in the level of detail provided, it seems that the model tends to provide longer answers in general. On the other hand, in example 3, the model appears to perform poorly on cases of subtle confusion since the model is unable to determine that the response doesn't quite match the question. In example 4, the model incorrectly predicts that the response contains a correct answer. Although the model appears to be correct, one could argue that the model should learn to reject answers that are not deducible from image features alone. In practice, this issue was not handled consistently by the annotators, so the model is often confused by these cases. Finally, in example 5, the model appears to be correct relative to the question and response, but in reality, the original image depicts a dog in a field. This is one of the relatively rare cases where the original image is necessary to accurately identify confusion, so the QR model makes an incorrect prediction.

7 Conclusion

In this paper, we describe two deep learning approaches for evaluating the quality of crowdsourced data labels and extracting correct answers from unstructured response text. To the best of our knowledge, this project marks the first time that a model has been designed to identify confusion in this way. Results show that our customizations of Pythia are effective in addressing the VQR task. We were able to design a model that could effectively identify confusion in responses (AUC-ROC = 0.79) and extract answers (F1=0.46).

Our second task, QR, was motivated by our initial finding that we, as humans, could often detect confusion in the responses from text features alone. The QR model implements a customized version of the BERT architecture to identify confusion solely from question and response text. Surprisingly, QR results are slightly better than those of the VQR models, which were provided with image features. Furthermore, the multi-task QR model showed better performance than the single-task model, achieving an an AUC-ROC score of 0.84 on the binary classification task and an F1 score of 0.78 on answer prediction. Thus, since extraction and analysis of image features is an extremely compute-intensive task, the QR model can be utilized to identify crowdworker confusion in resource-constrained settings, helping accelerate deep learning research.

A key limitation of this work is the presence of noisy ground truth answer labels, which likely influenced model performance. As discussed in section 3, ground truth answers were manually labeled by annotators, who examined the image, question, and user response to extract the true correct answer. We noticed several discrepancies in labels. For example, when questions were impossible to answer based on the image, some users still managed to provide a correct response; in these cases, annotators were often inconsistent as to whether or not the response should be regarded as valid. In addition, since the VQR dataset was constructed to train visual deep learning models to identify relevant features in images, questions were primarily focused on salient objects appearing in the image. However, many users misinterpreted questions and responded with irrelevant details, which were helpful to humans but not to the model.[5]

One future direction of this work is to train the model to identify such examples by determining when respondents provide information unrelated to image features. Another avenue is to use this data to improve the generation of the questions themselves. We could use our models to identify the types of questions likely to confuse crowdworkers; by adding a "possibility for confusion" feature to the question-generation model, we could encourage the original bot to ask better questions.

Overall, we believe that the techniques described in this project hold great potential for improving deep learning workflows by enabling automated quality evaluation of natural language data.

[5]For example, one user who posted a picture of their cat was asked "Where is the cat?"; an appropriate response would have provided location-specific information, such as "on the sofa." However, the user responded with "the cat died in 2016." To further compound the problem, manual annotators wrote the answer was "dead" for this example, rather than marking the response as confused.

8 Additional Information

Our mentor for this project is Sahil Chopra. We also have an external collaborator, Ranjay Krishna (from the Stanford AI Lab), who has graciously provided the dataset for this project.

References

[1] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6077–6086, 2018.

[2] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh. VQA: Visual Question Answering. In *International Conference on Computer Vision (ICCV)*, 2015.

[3] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[4] X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[6] B. Eisner, T. Rocktäschel, I. Augenstein, M. Bosnjak, and S. Riedel. emoji2vec: Learning emoji representations from their description. *CoRR*, abs/1609.08359, 2016.

[7] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He. Detectron. `https://github.com/facebookresearch/detectron`, 2018.

[8] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh. Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[9] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[10] J. Howard and S. Ruder. Universal language model fine-tuning for text classification, 2018.

[11] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[12] Y. Jiang, V. Natarajan, X. Chen, M. Rohrbach, D. Batra, and D. Parikh. Pythia. `https://github.com/facebookresearch/pythia`, 2018.

[13] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73, 2017.

[14] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[15] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[16] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018.

[17] P. Rajpurkar, R. Jia, and P. Liang. Know what you don't know: Unanswerable questions for squad. *CoRR*, abs/1806.03822, 2018.

[18] Yu Jiang*, Vivek Natarajan*, Xinlei Chen*, M. Rohrbach, D. Batra, and D. Parikh. Pythia v0.1: the winning entry to the vqa challenge 2018. *arXiv preprint arXiv:1807.09956*, 2018.

[19] P. Zhang, Y. Goyal, D. Summers-Stay, D. Batra, and D. Parikh. Yin and Yang: Balancing and answering binary visual questions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

9 Appendix

Group	Correct Answer (0)	Confusion (1)
Train	26,153	14,496
Validation	3242	1795
Test	3169	1773
Total	32,564	18,064

Table 4: Split breakdown by label.