Appendix

Abstract

*In our work we presented the new task of Visual Commonsense Reasoning and introduced a large-scale dataset for the task, VCR, along with Adversarial Matching, the machinery that made the dataset construction possible. We also presented **R2C**, a new model for the task. In the supplemental material, we provide the following items that shed further insight on these contributions:*

- *Additional dataset analysis (Section A)*
- *More information about dataset creation (Section B) and Adversarial Matching (Section C)*
- *An extended discussion on language priors (Section D)*
- *Model hyperparameters used (Section E)*
- *Additional VQA Baseline Results, with BERT embeddings (Section F)*
- *A datasheet for VCR (Section G)*
- *A visualization of **R2C**'s predictions (Section H)*

For more examples, and to obtain the dataset and code, check out visualcommonsense.com.

A. Dataset Analysis

In this section, we continue our high-level analysis of VCR.

A.1. Language complexity and diversity

How challenging is the language in VCR? We show several statistics in Table 3. Of note, unlike many question-answering datasets wherein the answer is a single word, our answers average to more than 7.5 words. The rationales are even longer, averaging at more than 16 words.

An additional informative statistic is the counts of unique answers and rationales in the dataset, which we plot in Figure 7. As shown, almost every answer and rationale is unique.

A.2. Objects covered

On average, there are roughly two objects mentioned over a question, answer, and rationale. Most of these objects are people (Figure 8), though other types of COCO objects are common too [49]. Objects such as 'chair,' 'tie,' and 'cup' are often detected, however, these objects vary in terms of scene importance: even though more ties exist in the data than cars, workers refer to cars more in their questions, answers, and rationales. Some objects, such as hair driers and snowboards, are rarely detected.

	Train	Val	Test
Number of questions	212,923	26,534	25,263
Number of answers per question	4	4	4
Number of rationales per question	4	4	4
Number of images	80,418	9,929	9,557
Number of movies covered	1,945	244	189
Average question length	6.61	6.63	6.58
Average answer length	7.54	7.65	7.55
Average rationale length	16.16	16.19	16.07
Average # of objects mentioned	1.84	1.85	1.82

Table 3: High level dataset statistics, split by fold (train, validation, and test). Note that we held out one fold in the dataset for blind evaluation at a later date; this fold is blind to us to preserve the integrity of the held-out data. Accordingly, the statistics of that fold are not represented here.

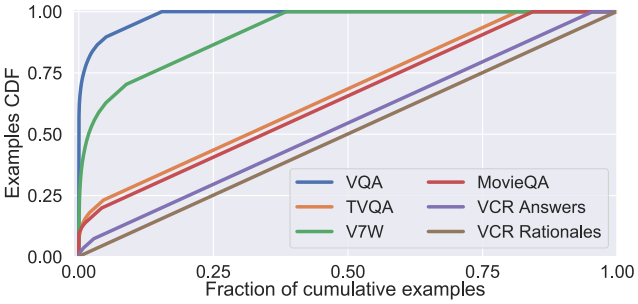


Figure 7: CDF of dataset examples ordered by frequency in question-answering datasets [5, 93, 75, 46]. To obtain this plot, we sampled 10,000 answers from each dataset (or rationales, for 'VCR rationales'). We consider two examples to be the same if they exactly match, after tokenization, lemmatization, and removal of stopwords. Where many datasets in this space are light-tailed, our dataset shows great diversity (e.g. almost every rationale is unique.)

A.3. Movies covered

Our dataset also covers a broad range of movies - over 2000 in all, mostly via MovieClips (Figure 9). We note that since we split the dataset by movie, the validation and test sets cover a completely disjoint set of movies, which forces a model to generalize. For each movie image, workers ask 2.6 questions on average (Figure 10), though the exact number varies - by design, workers ask more questions for more interesting images.

A.4. Inference types

It is challenging to accurately categorize commonsense and cognition-level phenomena in the dataset. One approach that we presented in Figure 2 is to categorize questions by type: to estimate this over the entire training set, we used a several patterns, which we show in Table 4. Still,

Type	Freq.	Patterns
Explanation	38%	why, how come, how does
Activity	24%	doing, looking, event, playing, preparing
Temporal	13%	happened, before, after, earlier, later, next
Mental	8%	feeling, thinking, saying, love, upset, angry
Role	7%	relation, occupation, strangers, married
Scene	5%	where, time, near
Hypothetical	5%	if, would, could, chance, might, may

Table 4: Some of the rules we used to determine the type of each question. Any question containing a word from one of the above groups (such as 'why') was determined to be of that type ('explanation').

we note that automatic categorization of the inference types required for this task is hard. This is in part because a single question might require multiple types of reasoning: for example, 'Why does `person1` feel embarrassed?' requires reasoning about `person1`'s mental state, as well as requiring an explanation. For this reason, we argue that this breakdown underestimates the task difficulty.

B. Dataset Creation Details

In this section, we elaborate more on how we collected VCR, and about our crowdsourcing process.

B.1. Shot detection pipeline

The images in VCR are extracted from video clips from LSMDC [67] and MovieClips. These clips vary in length from a few seconds (LSMDC) to several minutes (MovieClips). Thus, to obtain more still images from these clips, we performed shot detection. Our pipeline is as follows:

- We iterate through a video clip at a speed of one frame per second.
- During each iteration, we also perform shot detection: if we detect a mean difference of 30 pixels in HSV space, then we register a shot boundary.
- After a shot boundary is found, we apply Mask-RCNN [29, 24] on the middle frame for the shot, and save the resulting image and detection information.

We used a threshold of 0.7 for Mask-RCNN, and the best detection/segmentation model available for us at the time: X-101-64x4d-FPN[14], which obtains 42.4 box mAP on COCO, and 37.5 mask mAP.

B.2. Interestingness Filter

Recall that we use an 'interestingness filter' to ensure that the images in our dataset are high quality. First, every image had to have at least two people in it, as detected by Mask RCNN. However, we also found that many images with two or more people were still not very interesting. The two main failure cases here are when there are one or two

[14] Available via the Detectron Model Zoo.

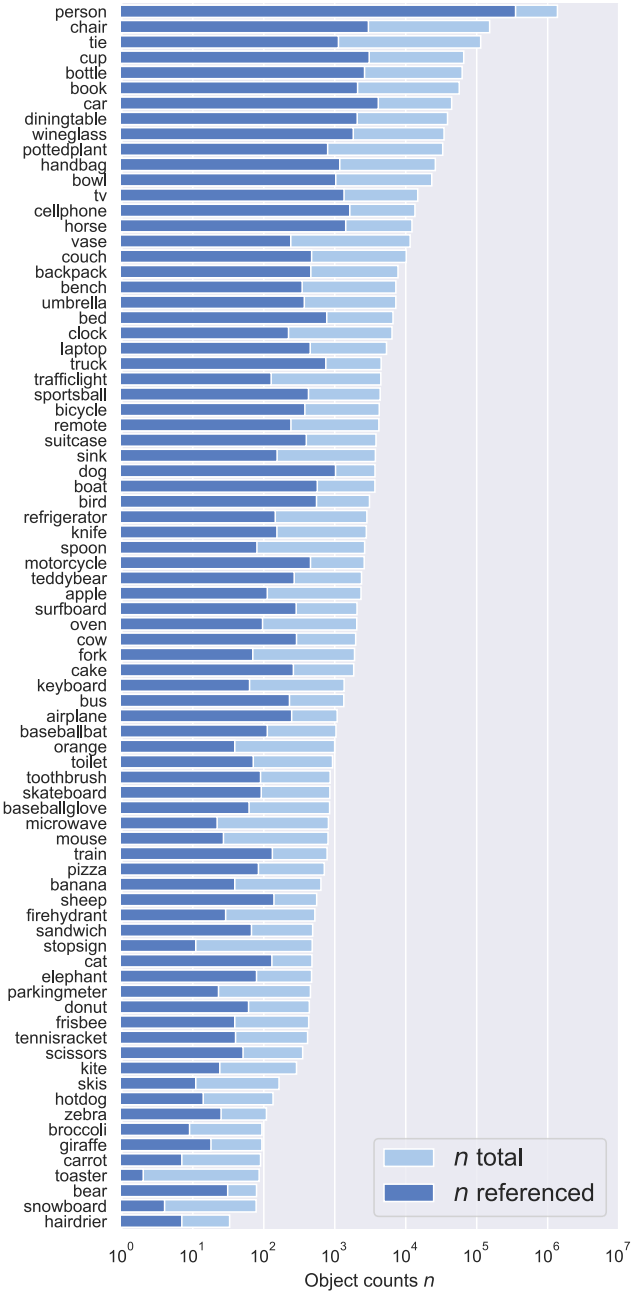


Figure 8: Distribution of the referenced COCO [49] objects in VCR. We count an object as being 'referenced' if, for a given question, answer, and rationale, that object is mentioned explicitly. Note that we do not double-count objects here - if `person5` is mentioned in the question and the answer, we count it once. This chart suggests that our dataset is mostly human-centric, with some categories being referenced more than others (cars are mentioned more than ties, even though cars appear less often).

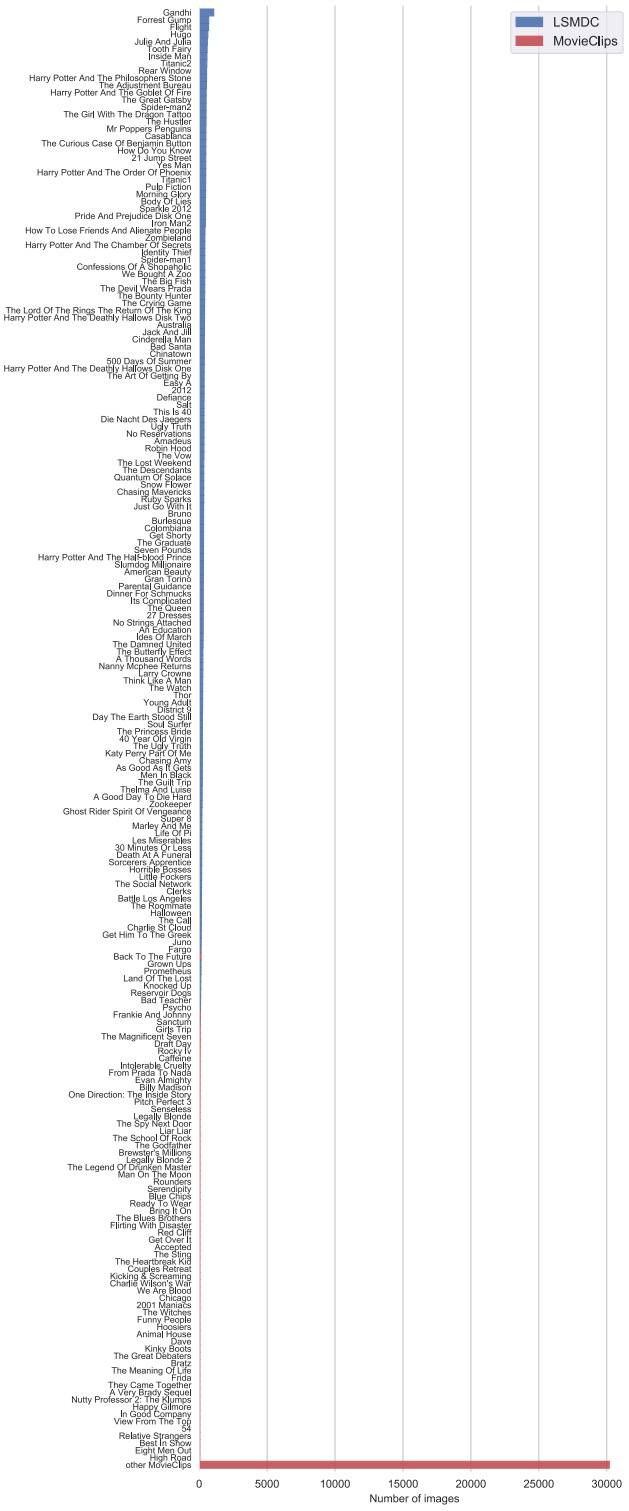


Figure 9: Distribution of movies in the VCR training set by number of images. Blue bars are movies from LSMDC (46k images); red are MovieClips (33k images). The MovieClips images are spread over a wider range of movies: due to space restrictions, most are under 'other MovieClips.'

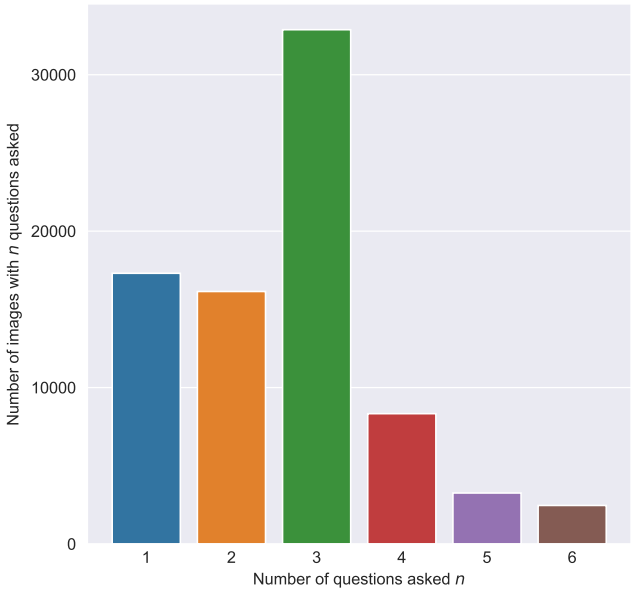


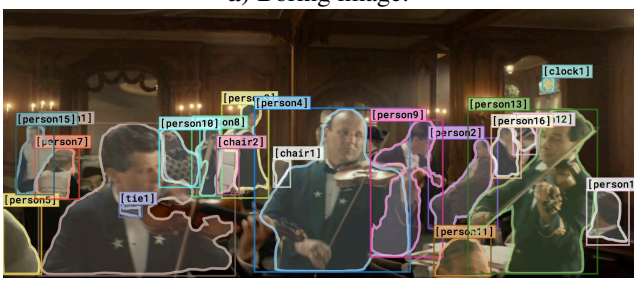
Figure 10: Number of questions asked per image on the VCR training set. The average number of questions asked per image is 2.645. Note that while workers could ask anywhere between one to three questions per image, images that were flagged as especially interesting by workers got re-annotated with additional annotations.

people detected, but they aren't doing anything interesting (Figure 11a), or when the image is especially grainy and blurry. Thus, we opted to learn an additional classifier for determining which images were interesting.

Our filtering process evolved as we collected data for the task. The first author of this paper first manually annotated 2000 images from LSMDC [67] as being 'interesting' or 'not interesting' and trained a logistic regression model to predict said label. The model is given as input the number of people detected by Mask RCNN [29, 24], along with the number of objects (that are not people) detected. We used this model to identify interesting images in LSMDC, using a threshold that corresponded to 70% precision. This resulted in 72k images selected; these images were annotated first.

During the crowdsourcing process, we obtained data that allowed us to build an even better interestingness filter later on. Workers were asked, along with each image, whether they thought that the image was especially interesting (and thus should go to more workers), just okay, or especially boring (and hard to ask even one good question for). We used this to train a deeper model for this task. The model uses a ResNet 50 backbone over the entire image [30] as well as a multilayer perceptron over the object counts. The entire model is trained end-to-end: 2048 dimensional features from Resnet are concatenated with a 512 dimensional projetion of the object counts, and used to predict the la-

a) Boring image.



b) Interesting image.

Figure 11: Two example images that come from the raw video pipeline. Image a) is flagged by our initial filter as 'boring', because there are only two people without any additional objects, whereas image b) is flagged as being interesting due to the number of people and objects detected.

bels.[15] We used this model to select the most interesting 40k images from Movieclips, which finished off the annotation process.

B.3. Crowdsourcing quality data

As mentioned in the paper, crowdsourcing data at the quality and scale of VCR is challenging. We used several best practices for crowdsourcing, which we elaborate on in this section.

We used Amazon Mechanical Turk for our crowdsourcing. A screenshot of our interface is given in Figure 12. Given an image, workers asked questions, answered them, and provided a rationale explaining why their answer might be correct. These are all written in a mixture of natural language text, as well as referring to detection regions. In our annotation UI, workers refer to the regions by writing the tag number.[16]

Workers could ask anywhere between one to three questions per HIT. We paid the workers proportionally at $0.22 per triplet. According to workers, this resulted in $8–25/hr. This proved necessary as workers reported feeling

[15] In addition to predicting interestingness, the model also predicts the number of questions a worker asks, but we never ended up using these predictions.

[16] Note that this differs a bit from the format in the paper: we originally had workers write out the full tag, like `[person5]`, but this is often long and the workers would sometimes forget the brackets. Thus, the tag format here is just a single number, like `5`.

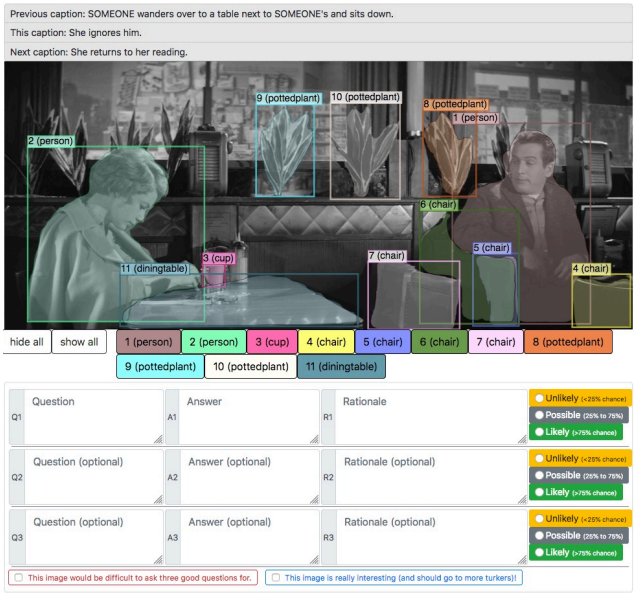


Figure 12: Screenshot of our annotation interface. Workers are given an image, as well as context from the video (here, captions from LSMDC [67]), and are asked to write one to three questions, answers, and rationales. For each answer, they must mark it as likely, possible, or unlikely. Workers also select whether the image was especially interesting or boring, as this allows us to train a deep model for predicting image interestingness.

"drained" by the high quality required.

Automated quality checks We added several automated checks to the crowdsourcing UI to ensure high quality. The workers had to write at least four words for the question, three for the answer, and five for the rationale. Additionally, the workers had to explicitly refer to at least one detection on average per question, answer, and rationale triplet. This was automatically detected to ensure that the workers were referring to the detection tags in their submissions.

We also noticed early on was that sometimes workers would write detailed stories that were only loosely connected with the semantic content of the image. To fix this, workers also had to self-report whether their answer was likely (above 75% probability), possible (25-75% probability), or unlikely (below 25% probability). We found that this helped deter workers from coming up with consistently unlikely answers for each image. The likelihood ratings were never used for the task, since we found they weren't necessary to obtain high human agreement.

Instructions Like for any crowdsourcing task, we found wording the instructions carefully to be crucial. We encouraged workers to ask about higher-level actions, versus lower-level ones (such as 'What is `person1` wearing?'), as well as to not ask questions and answers that were overly

generic (and thus could apply to many images). Workers were encouraged to answer reasonably in a way that was not overly unlikely or unreasonable. To this end, we provided the workers with high-quality example questions, answers, and rationales.

Qualification exam Since we were picky about the types of questions asked, and the format of the answers and rationales, workers had to pass a qualification task to double check that they understood the format. The qualification test included a mix of multiple-choice graded answers as well as a short written section, which was to provide a single question, answer, and rationale for an image. The written answer was checked manually by the first author of this paper.

Work verification In addition to the initial qualification exam, we also periodically monitored the annotation quality. Every 48 hours, the first author of this paper would review work and provide aggregate feedback to ensure that workers were asking good questions, answering them well, and structuring the rationales in the right way. Because this took significant time, we then selected several outstanding workers and paid them to do this job for us: through a separate set of HITs, these outstanding workers were paid $0.40 to provide detailed feedback on a submission that another worker made. Roughly one in fifty HITs were annotated in this way to give extra feedback. Throughout this process, workers whose submission quality dropped were dequalified from the HITs.

C. Adversarial Matching Details

There are a few more details that we found useful when performing the Adversarial Matching to create VCR, which we discuss in this section.

Aligning Detections In practice, most responses in our dataset are not relevant to most queries, due to the diversity of responses in our dataset and the range of detection tags (`person1`, etc.).

To fix this, for each query $\boldsymbol{q}_i$ (with associated object list $\boldsymbol{o}_i$ and response $\boldsymbol{r}_i$) we turn each candidate $\boldsymbol{r}_j$ into a template, and use a rule based system to probabilistically remap its detection tags to match the objects in $\boldsymbol{o}_i$. With some probability, a tag in $\boldsymbol{r}_j$ is replaced with a tag in $\boldsymbol{q}_i$ and $\boldsymbol{r}_i$. Otherwise, it is replaced with a random tag from $\boldsymbol{o}_i$.

We note that our approach isn't perfect. The remapping system often produces responses that violate predicate/argument structure, such as '`person1` is kissing `person1`.' However, *our approach does not need to be perfect*: because the detections for response $\boldsymbol{r}_j$ are remapped uniquely for each query $\boldsymbol{q}_i$, with some probability, there should be at least some remappings of $\boldsymbol{r}_i$ that make sense, and the question relevance model P_{rel} should select them.

Semantic categories Recall that we use 11 folds for the dataset of around 290k questions, answers, and rationales. Since we must perform Adversarial Matching once for the answers, as well as for the rationales, this would naively involve 22 matchings on a fold size of roughly 26k. We found that the major computational bottleneck wasn't the bipartite matching[17], but rather the computation of all-pairs similarity and relevance between ~26k examples.

There is one additional potential problem: we want the dataset examples to require a lot of complex commonsense reasoning, rather than simple attribute identification. However, if the response and the query disagree in terms of gender pronouns, then many of the dataset examples can be reduced to gender identification.

We address both of these problems by dividing each fold into 'buckets' of 3k examples for matching. We divide the examples up in terms of the pronouns in the response: if the response contains a female or male pronoun, then we put the example into a 'female' or 'male' bucket, respectively, otherwise the response goes into the 'neutral' bucket. To further divide the dataset examples, we also put different question types in different buckets for the question answering task (e.g. who, what, etc.). For the answer justification task, we cluster the questions and answers using their average GloVe embeddings [56].

Relevance model details Recall that our relevance model P_{rel} is trained to predict the probability that a response $\boldsymbol{r}$ is valid for a query $\boldsymbol{q}$. We used BERT for this task [15], as it achieves state-of-the-art results across many two-sentence inference tasks. Each input looks like the following, where the query and response are concatenated with a separator in between:

```
[CLS] what is casey doing ?  [SEP] casey is
getting out of car .  [SEP]
```

Note that in the above example, object tags are replaced with the class name (`car3`→`car`). Person tags are replaced with gender neutral names (`person1`→`casey`) [19].

We fine-tune BERT by treating it as a two-way classification problem. With probability 25% for a query, BERT is given that query's actual response, otherwise it is given a random response (where the detections were remapped). Then, the model must predict whether it was given the actual response or not. We used a learning rate of $2 \cdot 10^{-5}$, the Adam optimizer [44], a batch size of 32, and 3 epochs of fine-tuning.[18]

Due to computational limitations, we used BERT-Base as the architecture rather than BERT-Large - the latter is significantly slower.[19] Already, P_{rel} has an immense computational requirement as it must compute all-pairs simi-

[17] We use the https://github.com/gatagat/lap implementation.

[18] We note that during the Adversarial Matching process, for either Question Answering or Answer Justification, the dataset is broken up into 11 folds. For each fold, BERT is fine-tuned on the other folds, not on the final dataset splits.

[19] Also, BERT-Large requires much more memory, enough so that it's harder to fine-tune due to the smaller feasible batch size.

larity for the entire dataset, over buckets of 3000 examples. Thus, we opted to use a larger bucket size rather than a more expensive model.

Similarity model details While we want the responses to be highly relevant to the query, we also want to avoid cases where two responses might be conflated by humans - particularly when one is the correct response. This conflation might occur for several reasons: possibly, two responses are *paraphrases* of one another, or one response *entails* another. We lump both under the 'similarity' umbrella as mentioned in the paper and introduce a model, P_{sim}, to predict the probability of this occurring - broadly speaking, that two responses $\boldsymbol{r}_i$ and $\boldsymbol{r}_j$ have the same meaning.

We used ESIM+ELMo for this task [10, 57], as it still does quite well on two-sentence natural language inference tasks (although not as well as BERT), and can be made much more efficient. At test time, the model makes the similarity prediction when given two token sequences.[20]

We trained this model on freely available NLP corpora. We used the SNLI formalism [8], in which two sentences are an 'entailment' if the first entails the second, 'contradiction' if the first is contradicted by the second, and 'neutral' otherwise. We combined data from SNLI and MultiNLI [82] as training data. Additionally, we found that even after training on these corpora, the model would struggle with paraphrases, so we also translated SNLI sentences from English to German and back using the Nematus machine translation system [81, 73]. These sentences served as extra paraphrase data and were assigned the 'entailment' label. We also used randomly sampled sentence pairs from SNLI as additional 'neutral' training data. We held out the SNLI validation set to determine when to stop training. We used standard hyperparameters for ESIM+ELMo as given by the AllenNLP library [22].

Given the trained model P_{nli}, we defined the similarity model as the maximum entailment probability for either way of ordering the two responses:

$$P_{sim}(\boldsymbol{r}_i, \boldsymbol{r}_j) = \max\Big\{P_{nli}(\text{ent}|\boldsymbol{r}_i, \boldsymbol{r}_j), P_{nli}(\text{ent}|\boldsymbol{r}_j, \boldsymbol{r}_i)\Big\}, \quad (3)$$

where 'ent' refers to the 'entailment' label. If one response entails the other, we flag them as similar, even if the reverse entailment is not true, because such a response is likely to be a false positive as a distractor.

The benefit of using ESIM+ELMo for this task is that it can be made more efficient for the task of all-pairs sentence similarity. While much of the ESIM architecture involves computing attention between the two text sequences, everything before the first attention can be precomputed. This provides a large speedup, particularly as computing the ELMo representations is expensive. Now, for a fold size of N, we only have to compute $2N$ ELMo representations rather than N^2.

[20] Again, with object tags replaced with the class name, and person tags replaced by gender neutral names.

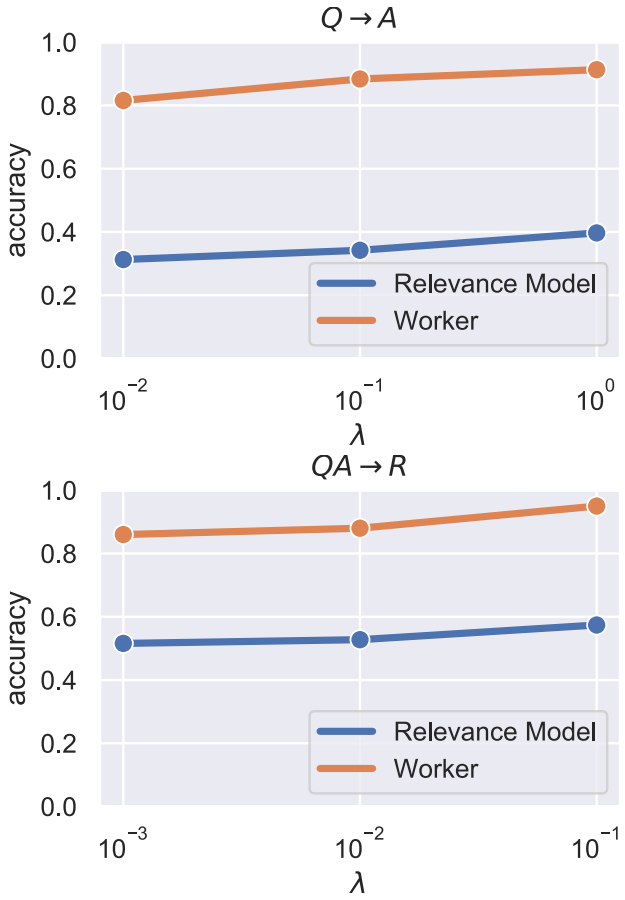


Figure 13: Tuning the λ hyperparameter. Workers were asked to solve 100 dataset examples from the validation set, as given by Adversarial Matching for each considered value of λ. We used these results to pick reasonable values for the hyperparameter such that the task was difficult for the question relevance model P_{rel}, while simple for human workers. We chose $\lambda = 0.1$ for $Q \rightarrow A$ and $\lambda = 0.01$ for $QA \rightarrow R$.

Validating the λ parameter Recall that our hyperparameter λ trades off between machine and human difficulty for our final dataset. We shed more insight on how we chose the exact value for λ in Figure 13. We tried several different values of λ and chose $\lambda = 0.1$ for $Q \rightarrow A$ and $\lambda = 0.01$ for $QA \rightarrow R$, as at these thresholds human performance was roughly 90%. For an easier dataset for both humans and machines, we would increase the hyperparameter.

D. Language Priors and Annotation Artifacts Discussion

There has been much research in the last few years in understanding what 'priors' datasets have.[21] Broadly speaking, how well do models do on VCR, as well as other visual question answering tasks, without vision?

To be more general, we will consider problems where a model is given a *question* and *answer choices*, and picks exactly one answer. The *answer choices* are the outputs that the model is deciding between (like the responses in VCR) and the *question* is the shared input that is common to all *answer choices* (the query, image, and detected objects in VCR). With this terminology, we can categorize unwanted dataset priors in the following ways:

- **Answer Priors**: A model can select a correct answer without even looking at the question. Many text-only datasets contain these priors. For instance, the RocStories dataset [53] (in which a model must classify endings to a story as correct or incorrect), a model can obtain 75% accuracy by looking at stylistic features (such as word choice and punctuation) in the endings.
- **Non-Visual Priors**: A model can select a correct answer using only non-visual elements of the question. One example is VQA 1.0 [5]: given a question like 'What color is the fire hydrant?' a model will classify some answers higher than others (red). This was addressed in VQA 2.0 [26], however, some answers will still be more likely than others (VQA's answers are open-ended, and an answer to 'What color is the fire hydrant?' must be a color).

These priors can either arise from biases in the world (fire hydrants are usually red), or, they can come from annotation artifacts [28]: patterns that arise when people write class-conditioned answers. Sometimes these biases are subliminal: when asked to write a correct or incorrect story ending, the correct endings tend to be longer [72]. Other cases are more obvious: workers often use patterns such as negation to write sentences that contradict a sentence [28].[22]

To what extent do vision datasets suffer from annotation artifacts, versus world priors? We narrow our focus to multiple-choice question answering datasets, in which for humans traditionally write correct *and* incorrect answers to a question (thus, potentially introducing the annotation artifacts). In Table 5 we consider several of these datasets: TVQA [46], containing video clips from TV shows, along with subtitles; MovieQA [75], with videos from movies and questions obtained from higher-level plot summaries; PororoQA [43], with cartoon videos; and TGIFQA [39], with templated questions from the TGIF dataset [47]. We note that these all differ from our proposed VCR in terms of subject matter, questions asked, number of answers (each of the above has 5 answers possible, while we have 4) and format; our focus here is to investigate how difficult these datasets are for text-only models.[23] Our point of comparison is VCR, since our use of Adversarial Matching means that humans never write incorrect answers.

Dataset	$\#_{train}$	Chance	A	Q+A	S+Q+A
TVQA [46]	122,039	20.0	45.0	47.4	70.6♠
MovieQA [75]	9,848	20.0	33.8	35.4	36.3♣
PororoQA [43]♡	7,530	20.0	43.1	47.4	
TGIFQA [39]♢	73,179	20.0	45.8	72.5	
VCR $Q \rightarrow A$	212,923	25.0	27.6	53.8	
VCR $QA \rightarrow R$		25.0	26.3	64.1	
VCR$^{\text{small}}$ $Q \rightarrow A$	9,848	25.0	25.5	39.9	
VCR$^{\text{small}}$ $QA \rightarrow R$		25.0	25.3	50.9	

Table 5: Text-only results on the validation sets of vision datasets, using BERT-Base. $\#_{train}$ shows the number of training examples. A corresponds to only seeing the answer; in Q+A the model also sees the question; in S+Q+A the model also sees subtitles from the video clip. These results suggest that many multiple choice QA datasets suffer from annotation artifacts, while Adversarial Matching helps produce a dataset with minimial biases; moreover, providing extra text-only information (like subtitles) greatly boosts performance. More info:

♠: State of the art.

♣: Only 45% (879/1958) of the questions in the MovieQA validation set have timestamps, which are needed to extract clip-level subtitles, so for the other 55%, we don't use any subtitle information.

♡: No official train/val/test split is available, so we split the data by movie, using 20% of data for validation and the rest for training.

♢: There seem to be issues with the publicly released train-test split of TGIFQA (namely, a model with high accuracy on a held-out part of the training set doesn't generalize to the provided test set) so we re-split the multiple-choice data ourselves by GIF and hold out 20% for validation.

We tackle this problem by running BERT-Base on these models [15]: given only the answer (A), the answer and the question (Q+A), or additional language context in the form of subtitles (S+Q+A), how well does BERT do? Our results in Table 5 help support our hypothesis regarding annotation

[21] This line of work is complementary to other notions of dataset bias, like understanding what phenomena datasets cover or don't [76], particularly how that relates to how marginalized groups are represented and portrayed [71, 90, 69, 68].

[22] For instance, the SNLI dataset contains pairs of sentences with labels such as 'entailed' or 'contradiction' [8]. For a sentence like 'A skateboarder is doing tricks' workers often write 'Nobody is doing tricks' which is a contradiction. The result is that the word 'nobody' is highly predictive of a word being a contradiction.

[23] It should be noted that all of these datasets were released before the existence of strong text-only baselines such as BERT.

artifacts: whereas accuracy on VCR, only given the ending, is 27% for $Q \rightarrow A$ and 26% for $Q \rightarrow A$, versus a 25% random baseline. Other models, where humans write the incorrect answers, have answer-only accuracies from 33.8% (MovieQA) to 45.8% (TGIFQA), over a 20% baseline.

There is also some non-visual bias for all datasets considered: from 35.4% when given the question and the answers (MovieQA) to 72.5% (TGIFQA). While these results suggest that MovieQA is incredibly difficult without seeing the video clip, there are two things to consider here. First, MovieQA is roughly 20x smaller than our dataset, with 9.8k examples in training. Thus, we also tried training BERT on 'VCR$^{\text{small}}$': taking 9.8k examples at random from our training set. Performance is roughly 14% worse, to the point of being roughly comparable to MovieQA.[24] Second, often times the examples in MovieQA have similar structure, which might help to alleviate stylistic priors, for example:

"Who has followed Boyle to Eamon's apartment?" Answers:

1. Thommo and his IRA squad.
2. Darren and his IRE squad.
3. Gary and his allies.
4. **Quinn and his IRA squad.**
5. Jimmy and his friends.

On the other hand, our dataset examples tend to be highly diverse in terms of syntax as well as high-level meaning, due to the similarity penalty. We hypothesize that this is why some language priors creep into VCR, particularly in the $QA \rightarrow R$ setting: given four very distinct rationales that ostensibly justify why an answer is true, some will likely serve as better justifications than others.

Furthermore, providing additional language information (such as subtitles) to a model tends to boost performance considerably. When given access to subtitles in TVQA,[25] BERT scores 70.6%, which to the best of our knowledge is a new state-of-the-art on TVQA.

In conclusion, dataset creation is highly difficult, particularly as there are many ways that unwanted bias can creep in during the dataset creation process. One such bias of this form includes annotation artifacts, which our analysis suggests is prevalent amongst multiple-choice VQA tasks wherein humans write the wrong endings. Our analysis also suggests Adversarial Matching can help minimize this effect, even when there are strong natural biases in the underlying textual data.

[24] Assuming an equal chance of choosing each incorrect ending, the results for BERT on an imaginary 4-answer version of TVQA and MovieQA would be 54.5% and 42.2%, respectively.

[25] We prepend the subtitles that are aligned to the video clip to the beginning of the question, with a special token (`;`) in between. We trim tokens from the subtitles when the total sequence length is above 128 tokens.

E. Model details

In this section, we discuss implementation details for our model, **R2C**.

BERT representations As mentioned in the paper, we used BERT to represent text [15]. We wanted to provide a fair comparison between our model and BERT, so we used BERT-Base for each. We tried to make our use of BERT to be as simple as possible, matching our use of it as a baseline. Given a query $\boldsymbol{q}$ and response choice $\boldsymbol{r}^{(i)}$, we merge both into a single sequence to give to BERT. One example might look like the following:

```
[CLS] why is riley riding motorcycle while
wearing a hospital gown ?  [SEP] she had to leave
the hospital in a hurry .  [SEP]
```

Note that in the above example, we replaced person tags with gender neutral names [19] (`person3`→ `riley`) and replaced object detections by their class name (`motorcycle1`→ `motorcycle`), to minimize domain shift between BERT's pretrained data (Wikipedia and the BookCorpus [94]) and VCR.

Each token in the sequence corresponds to a different transformer unit in BERT. We can then use the later layers in BERT to extract contextualized representations for the each token in the query (everything from `why` to `?`) and the response (`she` to `.`).[26] Note that this gives us a different representation for each response choice i.

We extract frozen BERT representations from the second-to-last layer of the Transformer.[27] Intuitively, this makes sense as the representations that that layer are used for both of BERT's pretraining tasks: next sentence prediction (the unit corresponding to the `[CLS]` token at the last layer L attends to all units at layer $L-1$), as well as masked language modeling (the unit for a word at layer L looks at its hidden state at the previous layer $L-1$, and uses that to attend to all other units as well). The experiments in [15] suggest that this works well, though not as well as fine-tuning BERT end-to-end or concatenating multiple layers of activations.[28] The tradeoff, however, is that precomputing BERT representations lets us substantially reduce the runtime of **R2C** and allows us to focus on learning more powerful vision representations.

Model Hyperparameters A more detailed discussion of the hyperparameters used for **R2C** is as follows. We tried

[26] The only slight difference is that, due to the WordPiece encoding scheme, rare words (like `chortled`) are broken up into subword units (`cho ##rt ##led`). In this case, we represent that word as the average of the BERT activations of its subwords.

[27] Since the domain that BERT was pretrained on (Wikipedia and the BookCorpus [94]) is still quite different from our domain, we fine-tuned BERT on the text of VCR (using the masked language modeling objective, as well as next sentence prediction) for one epoch to account for the domain shift, and then extracted the representations.

[28] This suggests, however, that if we also fine-tuned BERT along with the rest of the model parameters, the results of **R2C** would be higher.

to stick to simple settings (and when possible, used similar configurations for the baselines, particularly with respect to learning rates and hidden state sizes).

- Our projection of image features maps a 2176 dimensional hidden size (2048 from ResNet50 and 128 dimensional class embeddings) to a 512 dimensional vector.
- Our grounding LSTM is a single-layer bidirectional LSTM with a 1280-dimensional input size (768 from BERT and 512 from image features) and uses 256 dimensional hidden states.
- Our reasoning LSTM is a two-layer bidirectional LSTM with a 1536-dimensional input size (512 from image features, and 256 for each direction in the attended, grounded query and the grounded answer). It also uses 256-dimensional hidden states.
- The representation from the reasoning LSTM, grounded answer, and attended question is maxpooled and projected to a 1024-dimensional vector. That vector is used to predict the ith logit.
- For all LSTMs, we initialized the hidden-hidden weights using orthogonal initialization [70], and applied recurrent dropout to the LSTM input with $p_{drop} = 0.3$ [21].
- The Resnet50 backbone was pretrained on Imagenet [14, 30]. The parameters in the first three blocks of ResNet were frozen. The final block (after the RoiAlign is applied) is fine-tuned by our model. We were worried, however, that the these representations would drift and so we added an auxiliary loss to the model inspired by [48]: the 2048-dimensional representation of each object (without class embeddings) had to be predictive of that object's label (via a linear projection to the label space and a softmax).
- Often times, there are a lot of objects in the image that are not referred to by the query or response set. We filtered the objects considered by the model to include only the objects mentioned in the query and responses. We also passed in the entire image as an 'object' that the model could attend to in the object contextualization layer.
- We optimized **R2C** using Adam [44], with a learning rate of $2 \cdot 10^{-4}$ and weight decay of 10^{-4}. Our batch size was 96. We clipped the gradients to have a total L_2 norm of at most 1.0. We lowered the learning rate by a factor of 2 when we noticed a plateau (validation accuracy not increasing for two epochs in a row). Each model was trained for 20 epochs, which took roughly 20 hours over 3 NVIDIA Titan X GPUs.

	$Q \rightarrow A$		$QA \rightarrow R$		$Q \rightarrow AR$	
Model	GloVe	BERT	GloVe	BERT	GloVe	BERT
R2C	**46.4**	**63.8**	**38.3**	**67.2**	**18.3**	**43.1**
Revisited	39.4	57.5	34.0	63.5	13.5	36.8
BottomUp	42.8	62.3	25.1	63.0	10.7	39.6
MLB	45.5	61.8	36.1	65.4	17.0	40.6
MUTAN	44.4	61.0	32.0	64.4	14.1	39.3

Table 6: VQA baselines evaluated with GloVe or BERT, evaluated on the VCR evaluation set with **R2C** as comparison. While BERT helps the performance of these baselines, our model still performs the best in every setting.

F. VQA baselines with BERT

We present additional results where baselines for VQA [5] are augmented with BERT embeddings in Table 6. We didn't include these results in the main paper, because to the best of our knowledge prior work hasn't used contextualized representations for VQA. (Contextualized representations might be overkill, particularly as VQA questions are short and often simple). From the results, we find that while BERT also helps the baselines, our model **R2C** benefits even more, with a 2.5% overall boost in the holistic $Q \rightarrow AR$ setting.

G. VCR Datasheet

A datasheet is a list of questions that accompany datasets that are released, in part so that people think hard about the phenomena in their data [23]. In this section, we provide a datasheet for VCR.

G.1. Motivation for Dataset Creation

Why was the dataset created? The dataset was created to study the new task of Visual Commonsense Reasoning: essentially, to have models answer challenging cognition-level questions about images and also to choose a rationale justifying each answer.

Has the dataset been used already? Yes, at the time of writing, several groups have submitted models to our leaderboard at visualcommonsense.com/leaderboard.

Who funded the dataset?? VCR was funded via a variety of sources; the biggest sponsor was the IARPA DIVA program through D17PC00343.[29]

G.2. Dataset Composition

What are the instances? Each instance contains an image, a sequence of object regions and classes, a query, and a list of response choices. Exactly one response is correct. There are two sub-tasks to the dataset: in Question

[29] However, the views and conclusions contained herein are those of the authors and should not be interpreted as representing endorsements of IARPA, DOI/IBC, or the U.S. Government.

Answering ($Q \rightarrow A$) the query is a question and the response choices are answers. In Answer Justification ($QA \rightarrow R$) the query is a question and the correct answer; the responses are rationales that justify why someone would conclude that the answer is true. Both the query and the rationale refer to the objects using detection tags like `person1`.

How many instances are there? There are 212,923 training questions, 26,534 validation questions, and 25,263 questions. Each is associated with a four answer choices, and each question+correct answer is associated with four rationale choices.

What data does each instance consist of? The image from each instance comes from a movie, while the object detector was trained to detect objects in the COCO dataset [49]. Workers ask challenging high-level questions covering a wide variety of cognition-level phenomena. Then, workers provide a rationale: one to several sentences explaining how they came at their decision. The rationale points to details in the image, as well as background knowledge about how the world works. Each instance contains one correct answer and three incorrect counterfactual answers, along with one correct rationale and three incorrect rationales.

Does the data rely on external resources? No, everything is included.

Are there recommended data splits or evaluation measures? We release the training and validation sets, as well as the test set without labels. For the test set, researchers can submit their predictions to a public leaderboard. Evaluation is fairly straightforward as our task is multiple choice, but we will also release an evaluation script.

G.3. Data Collection Process

How was the data collected? We used movie images, with objects detected using Mask RCNN [24, 29]. We collected the questions, answers, and rationales on Amazon Mechanical Turk.

Who was involved in the collection process and what were their roles? We (the authors) did several rounds of pilot studies, and collected data at scale on Amazon Mechanical Turk. In the task, workers on Amazon Mechanical Turk could ask anywhere between one to three questions. For each question, they had to provide an answer, indicate its likelihood on an ordinal scale, and provide a rationale justifying why their answer is true. Workers were paid at 22 cents per question, answer, and rationale.

Over what time frame was the data collected? August to October 2018.

Does the dataset contain all possible instances? No. Visual Commonsense Inference is very broad, and we focused on a limited set of (interesting) phenomena. Beyond looking at different types of movies, or looking at the world beyond still photographs, there are also different types of inferences that we didn't cover in our work.

If the dataset is a sample, then what is the population? The population is that of movie images that were deemed interesting by our interestingness filter (having at least three object detections, of which at least two are people).

G.4. Data Preprocessing

What preprocessing was done? The line between data preprocessing and dataset collection is blurry for VCR. After obtaining crowdsourced questions, answers, and rationales, we applied Adversarial Matching, turning raw data into a multiple choice task. We also tokenized the text spans.

Was the raw data saved in addition to the cleaned data? Yes - the raw data is the correct answers (and as such is a subset of the 'cleaned' data).

Does this dataset collection/preprocessing procedure achieve the initial motivation? At this point, we think so. Our dataset is challenging for existing VQA systems, but easy for humans.

G.5. Dataset Distribution

How is the dataset distributed? VCR is freely available for research use at `visualcommonsense.com`.

G.6. Legal and Ethical Considerations

Were workers told what the dataset would be used for and did they consent? Yes - the instructions said that workers answers would be used in a dataset. We tried to be as upfront as possible to workers. Workers also consented to have their responses used in this way through the Amazon Mechanical Turk Participation Agreement.

If it relates to people, could this dataset expose people to harm or legal action? No - the questions, answers, and responses don't contain personal info about the crowd workers.

If it relates to people, does it unfairly advantage or disadvantage a particular social group? Unfortunately, movie data is highly biased against women and minorities [71, 69]. Our data, deriving from movies as well as from worker elicitations [68], is no different. For these reasons, we recommend that users do not deploy models trained on VCR in the real world.

H. Additional qualitative results

In this section, we present additional qualitative results from **R2C**. Our use of attention mechanisms allow us to better gain insight into how the model arrives at its decisions. In particular, the model uses the answer to attend

over the question, and it uses the answer to attend over relevant objects in the image. Looking at the attention maps help to visualize which items in the question are important (usually, the model focuses on the second half of the question, like 'covering his face' in Figure 14), as well as which objects are important (usually, the objects referred to by the answer are assigned the most weight).

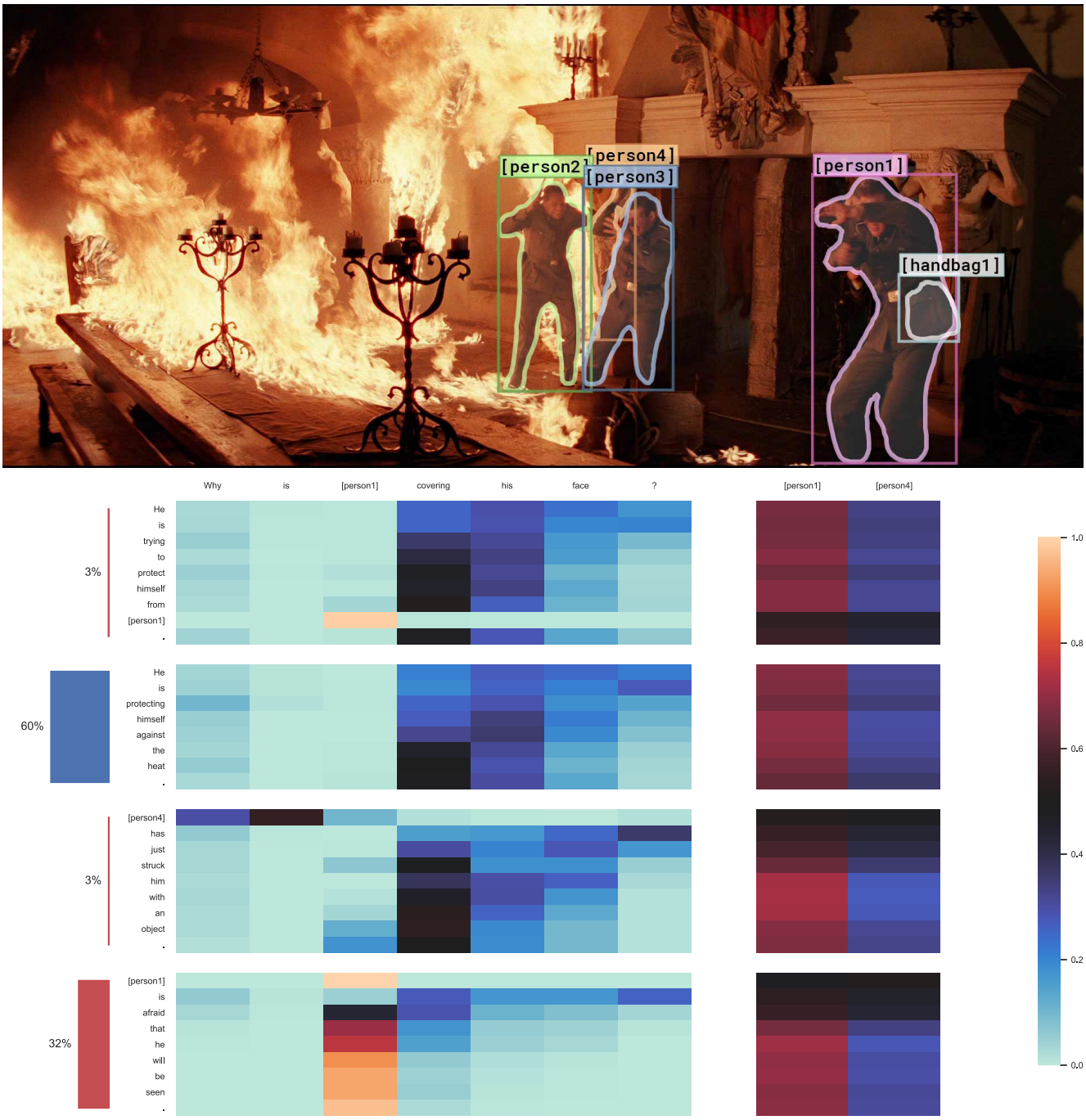


Figure 14: An example from the $Q \rightarrow A$ task. Each super-row is a response choice (four in total). The first super-column is the question: Here, 'Why is `[person1] covering his face?`' and the second super-column represents the relevant objects in the image that **R2C** attends to. Accordingly, each block is a heatmap of the attention between each response choice and the query, as well as each response choice and the objects. The final prediction is given by the bar graph on the left: The model is 60% confident that the right answer is **b.**, which is correct.

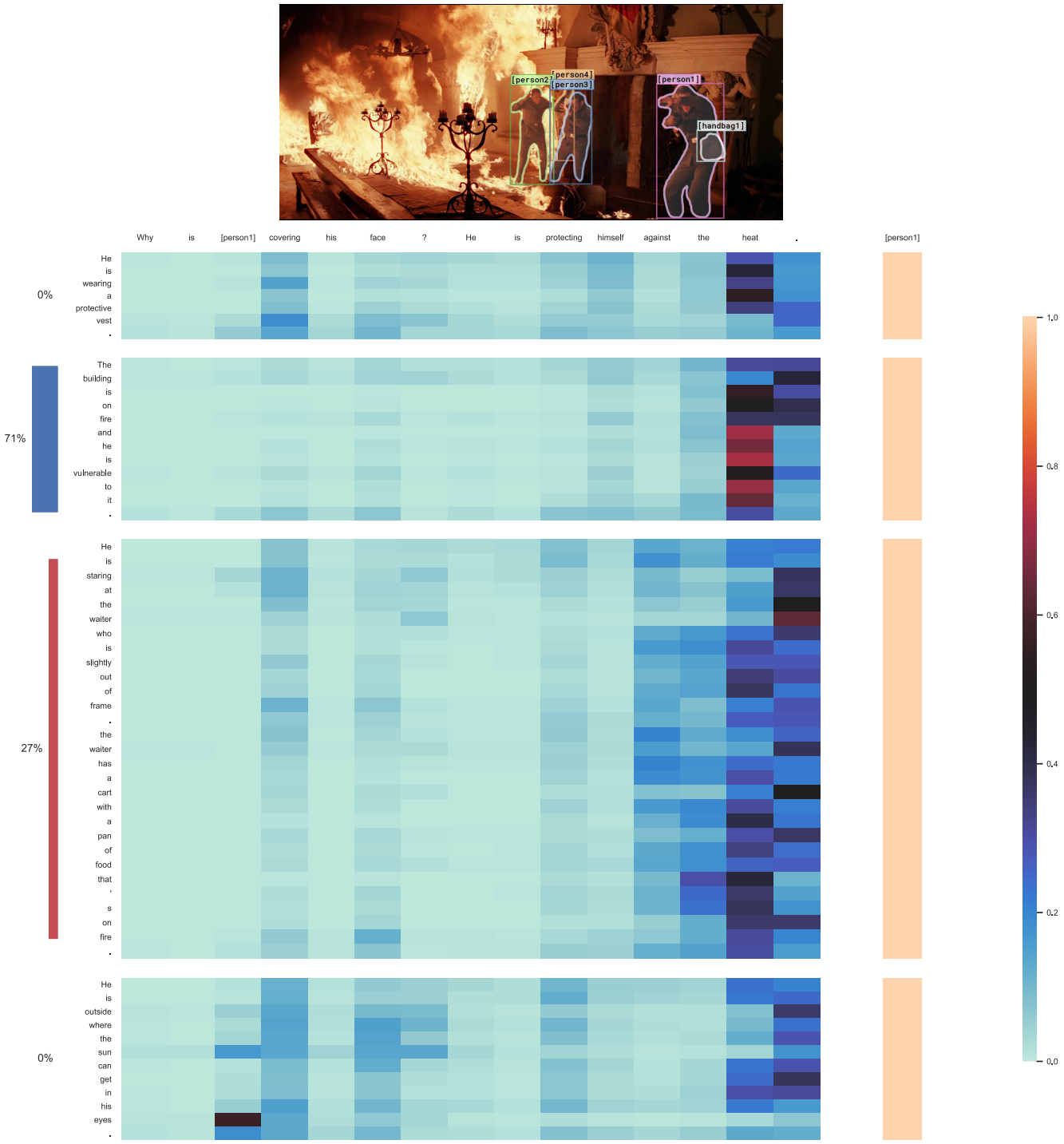


Figure 15: An example from the $QA \rightarrow R$ task. Each super-row is a response choice (four in total). The first super-column is the query, and the second super-column holds the relevant objects (here just a single person, as no other objects were mentioned by the query or responses). Each block is a heatmap of the attention between each response choice and the query, as well as the attention between each response choice and the objects. The final prediction is given by the bar graph on the left: The model is 71% confident that the right rationale is **b.**, which is correct.

Figure 16: An example from the $Q \rightarrow A$ task. Each super-row is a response choice (four in total). The first super-column is the question: Here, 'What is `[person13] doing?`' and the second super-column represents the relevant objects in the image that **R2C** attends to. Accordingly, each block is a heatmap of the attention between each response choice and the query, as well as each response choice and the objects. The final prediction is given by the bar graph on the left: The model is 86% confident that the right answer is **d.**, which is correct.

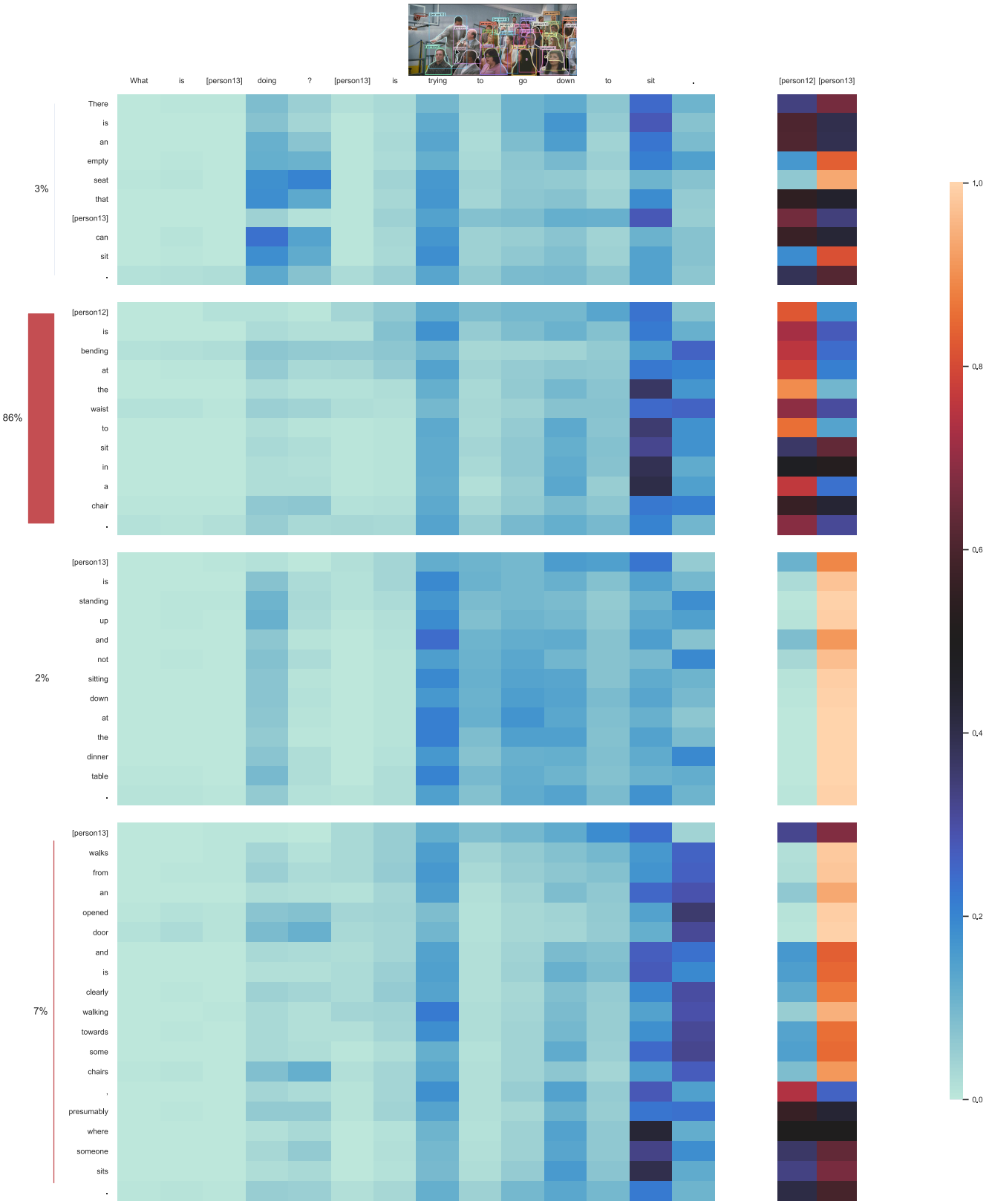


Figure 17: An example from the $QA \rightarrow R$ task. Each super-row is a response choice (four in total). The first super-column is the query, and the second super-column holds the relevant objects. Each block is a heatmap of the attention between each response choice and the query, as well as the attention between each response choice and the objects. The final prediction is given by the bar graph on the left: The model is 86% confident that the right rationale is b., which is incorrect - the correct choice is **a.**

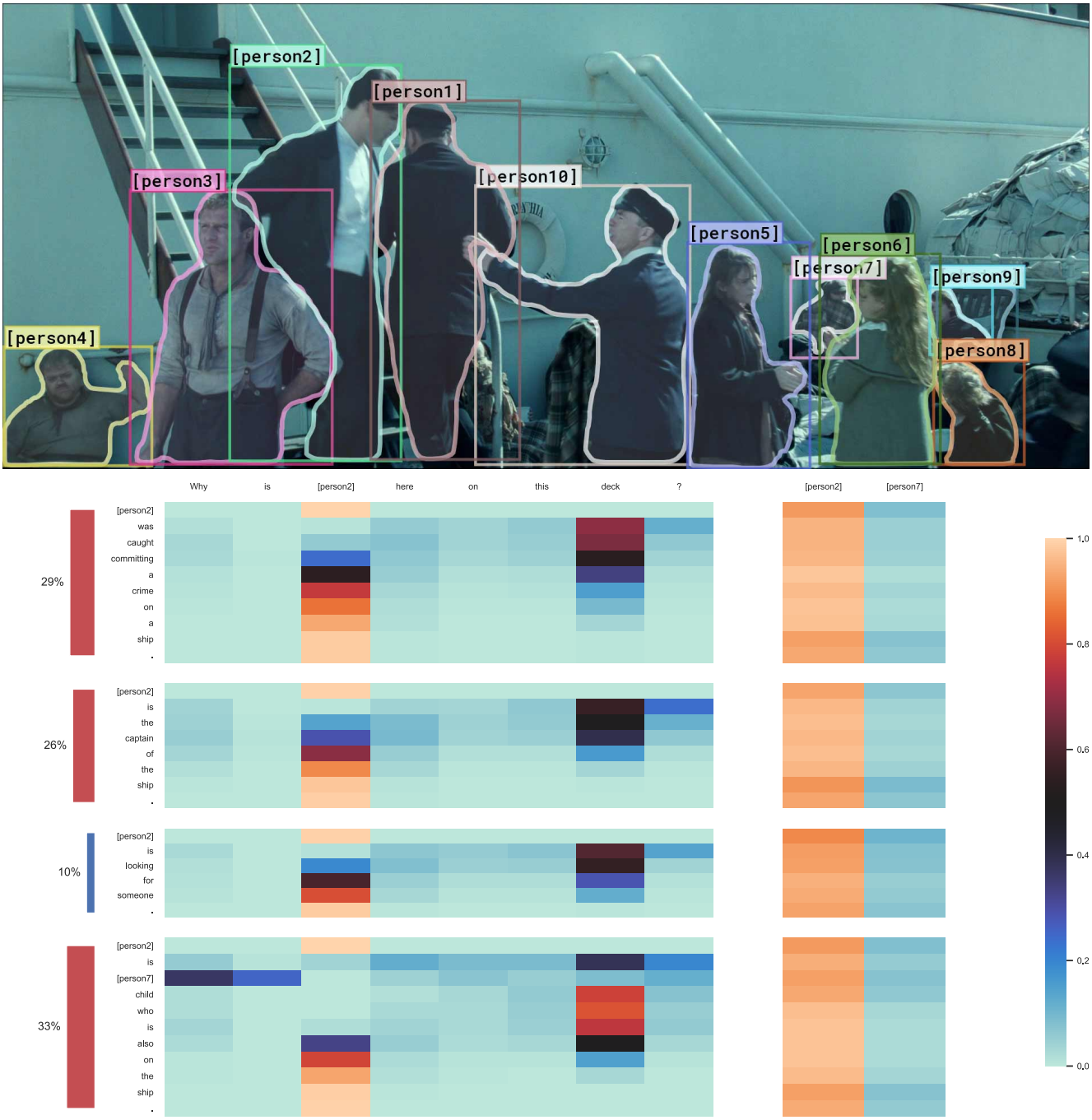


Figure 18: An example from the $Q \rightarrow A$ task. Each super-row is a response choice (four in total). The first super-column is the question: Here, 'Why is `[person2] here on this deck`?' and the second super-column represents the relevant objects in the image that **R2C** attends to. Accordingly, each block is a heatmap of the attention between each response choice and the query, as well as each response choice and the objects. The final prediction is given by the bar graph on the left: The model is 33% confident that the right answer is d., which is incorrect - the correct answer is correct answer is **c.**

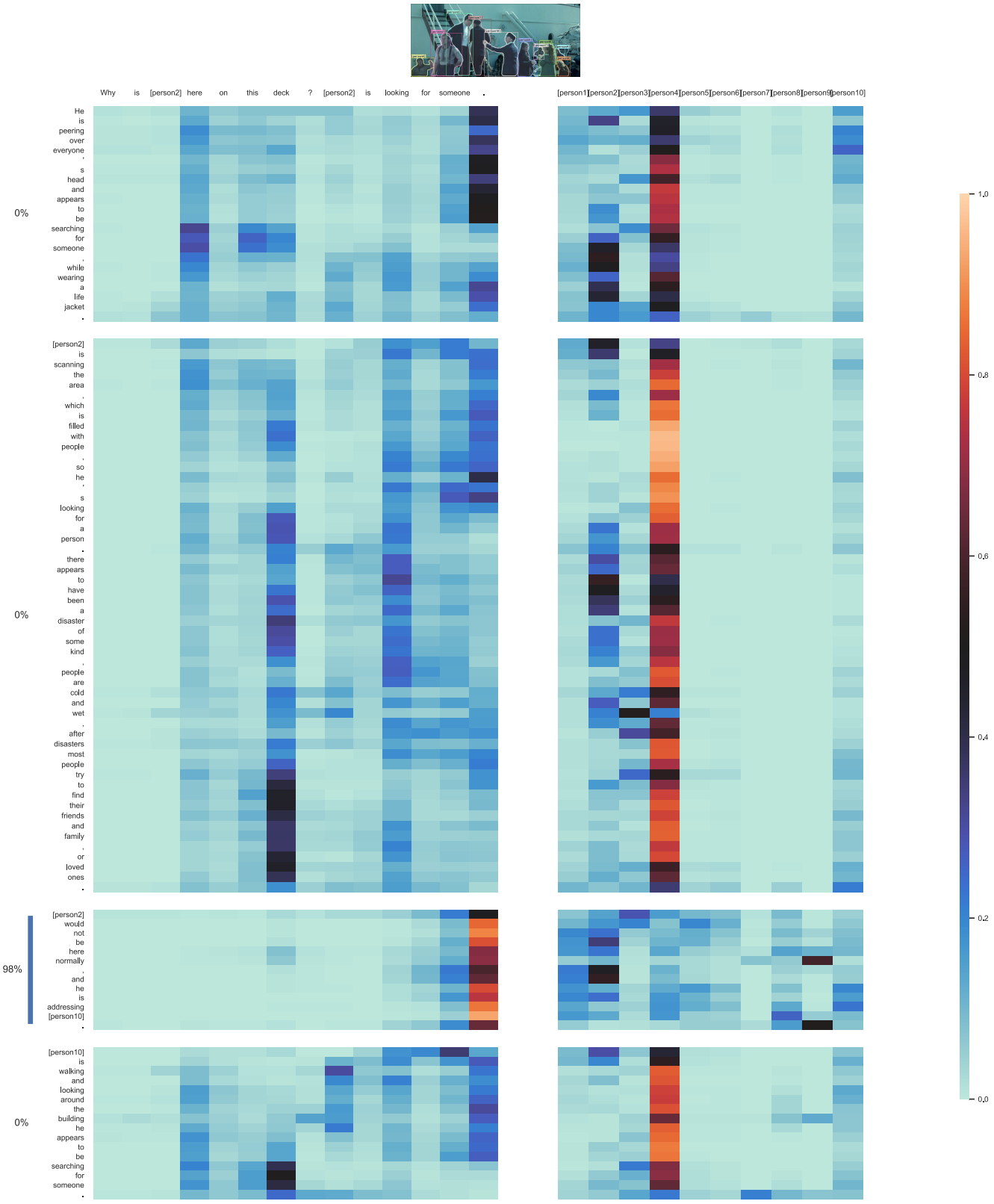


Figure 19: An example from the $QA \rightarrow R$ task. Each super-row is a response choice (four in total). The first super-column is the query, and the second super-column holds the relevant objects. Each block is a heatmap of the attention between each response choice and the query, as well as the attention between each response choice and the objects. The final prediction is given by the bar graph on the left: The model is 98% confident that the right rationale is **c.**, which is correct.