Greco et al. (2022) Replication Report

Atanasoska Tamara, and **Ryazanskaya Galina,** and **Verma Bhuvanesh**
University of Potsdam

Abstract

This paper presents the results of a partial replication experiment on Greco et al. (2020). The replication includes the experiments with blind and multimodal models on full dialogues, no last turn dialogues, and reversed dialogue history. We were able to run the LSTM, V-LSTM, and RoBERTa Encoder models. We were able to replicate the general findings of the paper, despite some discrepancies in the particular experiment results.

1 Introduction

The present paper reports the replication results of several experiments from Greco et al. (2020), "*Which Turn do Neural Models Exploit the Most to Solve GuessWhat? Diving into the Dialogue History Encoding in Transformers and LSTMs*". The original paper focuses on visually grounded dialogue history encoding. The authors use GuessWhat?! dataset as a test bed to compare the ability of various models to identify the salient information, as well as the relationship between this ability and the dialogue history processing order.

GuessWhat?! is an asymmetric visual dialogue task. In its complete form, the task requires one player, called Guesser, when presented with an image and a list of objects, to select the target object from the list by posing yes/no questions about the objects in the image and processing the replies given by the other player, called Oracle. The models used for this task usually compartmentalize the task into sub-tasks performed by sub-models: the Encoder (encoding of the image and dialogue history); the QGen (generating the questions); the Oracle (replying to the questions); and the Guesser (selecting the target object based on the encoding of the dialogue history and, in some cases, of the image; see Mogadala et al. (2021)).

It has previously been proposed, based on language-only datasets, that the neural models are insensitive to the order of turns in dialogues and might not be using the history effectively (Sankar et al., 2019). The authors try to assess whether this result applies to the multimodal dialogues as well. Therefore, the authors use a visual dialogue task and compare various encoders, assessing the contribution of the order of the dialogue history to the task success. They also claim that in the GuessWhat!? task the order of questions is not crucial, and that humans would be able to guess the target object even if the history was reversed. They wonder, whether the models would also have the ability to identify task-relevant information independently of the position. To single out the contribution of the Encoder differences, the authors keep all the other model components fixed, namely, by keeping the Guesser architecture the same across the experiments. Additionally, they use the human questions and replies from the dataset to ensure the dialogue quality, that is, to guarantee that the performance of the QGen and the Guesser does not affect the task success.

The Encoder models the authors compare vary along three dimensions: the architectural basis (Recurrent Neural Networks (RNN) vs Transformers); the input modality (language-only (blind) vs multimodal); and the utilization of the background model knowledge (models trained from scratch vs pre-trained and fine-tuned). The authors conduct multiple experiments with every model to identify the contribution of every turn to the task success and the ability of each model to identify salient information. They compare the models' performance on the complete unaltered dialogues, the dialogues with the last turn removed (across dialogue lengths), and the dialogues with reversed history. Additionally, the authors analyze the attention distribution by dialogue turn.

We decided to replicate some of the experiments from the paper, to test their reproducibility. Importantly, the original paper includes a dedicated section on details for reproducibility, therefore acknowledging its importance. Some of the code for the replication was available online[1]. Nevertheless, the stochastic nature of the neural models makes the reproduction a vital part of result verification. If the results are reproducible, they may be deemed reliable and the theoretical conclusions drawn from them more trustworthy.

We chose to replicate the experiments relating to two of the dimensions: model architecture and input modality. We chose to omit the experiment relating to the background model knowledge, as training Transformer models from scratch is computationally expensive and the results presented in the original paper indicated that the pre-trained and fine-tuned models mostly outperformed the ones trained from scratch. Unfortunately, due to a lack of available computational resources, we were unable to reproduce the LXMERT experiments. Additionally, we omitted the experiment related to the attention analysis due to the limited scope of the project.

Acknowledging the importance of reproducibility, we deem it necessary to provide the documented code that we used for the present project [2]. The repository follows the code provided by the authors of the original paper, supplemented by files created to simplify the training and testing, based on the existing code with minor alterations and bug-fixes. Additionally, our repository includes extended documentation to improve the usability.

2 Replication

2.1 Task Description

The original paper explores the influence of various Encoder models on GuessWhat?! task success and the attention paid by these models to different turns in the dialogue history. Therefore, the task is to train the relevant components of the GuessWhat?! model and to evaluate the resulting performance singling out the influence of the Encoders. To do so, the authors fix the non-encoder components, namely, they use the human questions and the answers from the dataset instead of QGen and Oracle, and assess the task success using the same Guesser architecture trained on the outputs of various tested Encoders. The Guesser architecture used is the one proposed in the original GuessWhat?! paper (De Vries et al., 2017).

As the authors were interested in the ability of each model to identify salient information, they explored to which turns the models pay attention and whether the order of turns impairs a model's performance. The particular tasks relating to this exploration included test set modifications. One modification was aimed at comparing the dialogues with and without the last turn, which, according to the authors contained a lot of task-relevant information and the drop in performance would therefore be indicative of the model's sensitivity to the salient information and the size of the drop could indicate the ability of the model to gather information from the remaining turns. The other experiment compared the performance on the unaltered dialogues to the performance on the reversed dialogues. Once more, the models were tested for the ability to identify the salient information independently of its dialogue position. Both dataset modifications are discussed in greater detail in the next section.

2.2 Datasets

The dataset the authors use is a subset of GuessWhat?! dataset collected by De Vries et al. (2017). The dataset consists of real-world images taken from the MS-COCO dataset (Lin et al., 2014) along with dialogues collected via Amazon Mechanical Turk. In the dialogues, one person is assigned the role of the Guesser, having to select the target object from the ones on the image by posing yes/no questions to the other player, assigned the role of the Oracle. The authors of the original paper filtered the dataset to only include the dialogues in which the Guesser successfully selected the target object and to only include the dialogues with 10 questions or less (90K training set, 18K both in validation & testing). The authors provide an in-depth analysis of the dataset features and their correlations (Greco et al., 2020).

Additionally, there were two test datasets created for the experiments. The first of the experimental datasets modified dialogue history, excluding the last turn of the dialogue. The motivation for this is that the last question is often long and includes many details, thus containing a lot of task-relevant information. The impact of the removal of the last turn on task success could show the model's sensitivity to the importance of the last turn.

The second experimental dataset included all the questions from the original dialogue but in reversed

[1] https://github.com/claudiogreco/aixia2021

[2] https://github.com/TamaraAtanasoska/dialogue-history

order. This modification of the dataset could allow the authors to test how much influence the correct dialogue order has on the task success. If the reversed order task success proves comparable to the original dataset, it would imply that the model is robust to the changes in turn order. The authors claim that a good Encoder should be able to identify the salient information independently of the position in the dialogue history.

2.3 Models

The experiments of the original article featured 4 models, each consisting of two parts: the Encoder, generating the representation of the dialogue history and, in some cases, of the image, and the Guesser, selecting the target from a list of candidate objects based on the Encoder output.

The Guesser architecture was the same across the models. The Guesser received as the input the category of each candidate object, its' spatial coordinates, and a representation of the dialogue history with an optional representation of the image, both obtained from the Encoder. The candidate object category and coordinates were then put through a Multi-Layer Perceptron (MLP). The MLP had three layers with 264, 512, and 512 nodes respectively. The embedding from the Encoder was dot-multiplied with the candidate object representations obtained from the MLP and put through softmax to obtain object probabilities.

As mentioned above, the Encoder models varied along two dimensions: input modality (blind vs multimodal) and base architecture (RNN vs Transformer). All of the Encoder models featured a linear layer with *tanh* activation that scaled the output of the encoder to the size of the Guesser input (512 nodes)[3].

The multimodal models required image preprocessing, specifically, the ResNet image features needed to be generated for all the datasets.

All the models were trained on a remote machine provided by the university, with 12G of GPU space (NVIDIA-SMI 510.54 with CUDA 11.6).

The next sections describe in detail the Encoder model used for the replication project.

[3]All the specific model hyperparameters are described in respective configuration files in the project repository at `https://github.com/TamaraAtanasoska/dialogue-history/tree/main/model-repos/aixia2021/config/SL`

2.3.1 Blind Models: LSTM

The LSTM Encoder architecture follows the one proposed in De Vries et al. (2017), encoding only the dialogue history using one layer of unidirectional Long-Short Term Memory units with 512 nodes.

The original article mentions using `tensorflow` for the LSTM model, implying the code for the De Vries et al. (2017)[4]. Therefore, we began by setting up and running the code in this repository. This included setting up the environment that would work with older `python` (3.6) and `tensorflow` (1.1.0) versions, as the code was last updated in 2018. Additionally, some bug-fixing was required to make the code run. The environment requirements, along with the commands to train and test the LSTM are documented in the repository. The bug-fixes were documented and suggested as a pull request to the original repository. We obtained practically the same results as the ones reported in the repository README. Yet, we saw that the model over-fitted after the tenth epoch. Moreover, the model trained quite slowly, which can be attributed to the fact that the code used older versions of machine learning libraries.

Fortunately, the paper-specific repository[5] featured a `PyTorch` version of the same LSTM Encoder, and we chose to use this version for the experiments. The LSTM model was trained for 30 epochs.

2.3.2 Blind Models: RoBERTa

The RoBERTa Encoder uses a modified BERT (Transformer) architecture proposed in Liu et al. (2019). We only used a pre-trained RoBERTa model (`roberta-base` provided in `transformers` library), having 12 self-attention layers with 12 heads each. The model was pre-trained for masked language modeling task for 500K steps on English text. The RoBERTa model was trained for 30 epochs.

2.3.3 Multimodal Models: V-LSTM

The multimodal representation for the LSTM-based architecture was achieved by concatenating the linguistic and visual representation and scaling its result with the MLP of the Guesser. The visual

[4]`https://github.com/GuessWhatGame/guesswhat`

[5]`https://github.com/claudiogreco/aixia2021`

representation was obtained using frozen ResNet-152 features. The files with the image features were provided by the authors of the original article, having 2048 visual features. The LSTM component consisted of one layer of unidirectional LSTM units with 1024 nodes. The V-LSTM model was trained for 30 epochs.

2.3.4 Multimodal Models: LXMERT

The transformer-based multimodal Encoder used LXMERT architecture Tan and Bansal (2019). This model represents the image features with position-aware object embeddings from the 36 most salient image regions identified using Faster R-CNN. The text is represented with position-aware word embeddings. Both modalities are processed by a Transformer-based architecture with self-attention (5 layers for visual and 9 for textual embedding). The two representations are combined using a multimodal transformer with cross-modal attention with 5 layers. The model was trained on a multi-objective multimodal task pool consisting of 5 tasks. The pre-trained model used for the experiments was obtained from the link provided in the original paper repository (Greco et al., 2020) [6]

The LXMERT model required the generation of special image features for Faster R-CNN, and the files required for the generation process were absent from the repository, as they were too large for GitHub. Upon request, the authors provided the image features, yet due to the limitations on available computational resources, we could not load the features. The direct attempts caused memory leaks, and the attempted fixes, such as removing multiprocessing or limiting memory usage, were unsuccessful in solving this problem. As a result, we had to leave the LXMERT outside the scope of the replication project.

2.4 Experiment 1: Task Success

The initial experiment to recreate from Greco et al. (2020) is the experiment aimed at comparison of the performance of various Encoders on Guess-What?! dataset. This included training the Guesser model with respective Encoders on the train set and evaluating their performance on the test set, with the test-valid-train split provided by the authors.

[6] `https://github.com/claudiogreco/aixia2021/tree/main/lxmert`
Importantly, it is not quite clear from the repository README if this is the original model provided by the authors of the original article, or the reproduction trained from scratch, and it is not quite clear what the model was trained on. Further reading and inquiries would be necessary to clarify that point.

	LSTM	V-LSTM	RoBERTa
original	64.7	64.5	67.9
present	65.3	65.0	68.7

Table 1: Test accuracy of the three Encoder models as reported in Greco et al. (2020) and in our replication experiment.

	LSTM	V-LSTM	RoBERTa
original	46.2 (18.5)	49.8 (14.7)	44.7 (23.2)
present	47.3 (18)	47.5 (17.5)	52.0 (16.7)

Table 2: Test accuracy of the three Encoder models on the dataset without the last turn as reported in Greco et al. (2020) and in our replication experiment. The numbers in parenthesis indicate the drop in performance as compared to the performance on the unaltered dialogues.

2.4.1 Replication Results: Task Success

The results of the replication experiment, as shown in Table 1, even though not matching the results reported by Greco et al. (2020) exactly, are very close to them (within 1%). Importantly, all the trends in the task success are preserved, that is, RoBERTa outperforms the two LSTM-based models, which perform very similarly. As the authors provided the random seed value in the configuration, the slight differences found might stem either from the difference in batch size (we had to make batches smaller to adjust for our limited computational resources) or from the differences in the way the random seed is handled by the `PyTorch`, as any slight difference in that could result in a different performance due to the stochastic nature of the models.

2.5 Experiment 2: No Last Turn

The experiment on the influence of the last turn on task success required running the models on the modified test set and comparing the results with the ones on the unaltered test set.

2.5.1 Replication Results: No Last Turn

Table 2 reports the performance of the models on the test set with the last turn removed for both the original and the replication results. In the replication results, RoBERTa performs the best and shows the least drop in performance, while the accuracy and the drop in accuracy of LSTM and V-LSTM are very similar. For LSTM and V-LSTM the perfor-

	LSTM	V-LSTM	RoBERTa
original	56 (8.7)	51.3 (13.2)	66.5 (1.4)
present	49.2 (16.1)	53.2 (11.8)	67.2 (1.5)

Table 3: Test accuracy of the three Encoder models on the dataset with the reversed dialogue history as reported in Greco et al. (2020) and in our replication experiment. The numbers in parenthesis indicate the drop in performance as compared to the performance on the unaltered dialogues.

mance appears quite similar between the original and the replication (with differences of 1.1% and -2.3% respectively), while in the performance of the RoBERTa there is a significant difference (the replication results being 7.3% better). Importantly, there are also differences in the size of the drop in performance (reported in parenthesis): while LSTM performance drops by the same 18% on the no-last-turn test set, for V-LSTM there is some difference in the size of the drop between the replication and the original (2.8%), and for RoBERTa the drop is significantly more pronounced in the original paper (the drop in performance that we observed in the replication being 6.5%).

While the overall performance trends for this experiment are quite similar, we observed a closer performance of LSTM and V-LSTM, and, interestingly, we observed RoBERTa still outperform the other models by some margin (4.5%), unlike the findings of the original experiment.

Here it is worth noting that in addition to the differences in batch size and possible changes due to random initialization, we are comparing the average of the results reported for the games of length 3, 5, and 8, to the results averaged across all the game lengths. This could the source of discrepancy in the performance patterns. Unfortunately, the result average across all lengths on the test set with the last turn removed is not reported in the original article.

2.6 Experiment 3: Reversed History

The experiment on the influence of the turn order inversion on task success required running the models on the modified test set and comparing the results with the ones on the unaltered test set.

2.6.1 Replication Results: Reversed History

Table 3 compares the original and the replicated model performance on the test set with the reversed dialogue history. Once again, RoBERTa is the best-performing model with the least drop in performance, followed by V-LSTM, and the LSTM in the replication experiment shows the worst performance and the largest drop in accuracy when the dialogue history is reversed, thus showing the least ability to identify salient information.

There are significant differences between the patterns observed in the original versus the replicated results: while RoBERTa performs very similarly in terms of absolute performance, the drop in performance, and compared to other models, the V-lSTM results are less similar (the difference being 1.9% in task success and -1.4% in performance drop), with LSTM showing the largest difference (6.8% in task success and 7.4% in performance drop).

It is not quite clear what could cause the differences in the results, beside the batch size and possible random initialization differences. One possible explanation for the large difference seen in the performance of LSTM model is that we chose to use the `PyTorch` code in the newer, paper-specific repository, over the older code in the De Vries et al. (2017) repository. The latter was reported in the README to have some bugs, which, if the authors used it, could have affected the LSTM results. However, in this case it is not clear why no large performance differences were observed in the other experiments.

2.7 Replication Challenges

There were a few challenges that we faced while carrying out the replication experiment, despite the reproducibility information provided in the paper and the availability of the code. They are be briefly outlined in this section.

To begin with, in both Greco et al. (2020) and De Vries et al. (2017) repositories there was some outdated, confusing and in some cases incorrect documentation, with the degrees varying from missing words and grammatical errors to referring to file locations not specified by the proposed file structure. The Greco et al. (2020) repository was missing the `requirement.txt` file, and we had to refer to a third repository [7] dedicated to GDSE to try to find the requirements. Even so, some requirements for both repositories had to be tested for compatibility with the specified ones and added manually.

Additionally, many of the commands and instruc-

[7] https://github.com/shekharRavi/Beyond-Task-Success-NAACL2019

tions were not accompanied by the explanations of their meaning, and at times they did not match anything reported in the article, leaving it for the reader to interpret why one needed these commands. For example, the blind models still required with image features. Some other features (crop/object features) were required in the code but unused. To find out the meaning and the use of the code we had to contact the authors and ask them for directions, and we were very lucky in that they had time to provide us with explanations.

There were also a few bugs and inconsistencies in the code provided in the repository. Firstly, all the code referred to a slightly different version of the dataset (the difference being in the JSON key names, resulting in a parsing error). Secondly, there were small bugs in things such as type casts, that prevented us from running the code right away. Thirdly, the image feature extraction using the provided script had an inconsistency, and we had to adapt the feature extraction script to produce correct features (required for further training).

The final group of challenges had to do with the specific machine we used to run the experiments, the limitations on computational resources, and difficulties in server data management. One problem had to do with the fact that we could not make code run on the available GPU machine because the `requirement.txt` asked for Pytorch 1.0 which runs on CUDA 10 or below, while we had CUDA 11+. Therefore, we had to migrate the code to the latest version of `PyTorch` to run the code with the available resources. Similarly, the De Vries et al. (2017) repository used `tensorflow` 1.1 and, crucially, `tensorflow-gpu`, which is no longer supported, and only available for python up to 3.6, which is also no longer supported on many machines. To account for that, we used virtual environments and documented them in the pull request to the repository for others to use. Finally, as we lacked the computational resources on the server to load some of the data, we had to spend time adapting the code, and we could still not run the LXMERT experiment. Additionally, the problems with limitations on memory caused memory leaks, killing all the processes on the server.

3 Conclusions

The replication experiment on the results discussed in Greco et al. (2020) reported in the present paper has been largely successful, as we were able to set up and run the code provided by the authors using most of their specifications. Secondly, we were able to replicate the trends in the results concerning the task success of the various encoder models on GuessWhat?! dataset. We observed, in line with the original findings, that the Transformer-based models (only RoBERTa in our case) outperformed the RNN-based models (LSTM and V-LSTM). The experiment on the role of the last turn has not been fully reproducible, as we saw RoBERTa outperform the other two models. The experiment on the inversion of the dialogue history, in line with the original findings, has shown that RoBERTa is the most robust and most able to extract the salient information independently of the position. Yet, we observed a reversed pattern with respect to the performance of LSTM and V-LSTM. Finally, we were able to reproduce the finding that blind (LSTM) and multimodal models (V-LSTM) did not differ strongly in their performance in any of the experiments. The largest difference, both in the original article and in our replication experiment was observed in the reversed dialogue experiment.

Here we would like to once again underline the importance of reproducibility and documentation of the scientific findings relating to neural networks, as the stochastic nature of these models deems reproduction much more important. Since the general findings of the original article, at least to the extent we could test, were indeed reproducible, they, along with the theoretical conclusions drawn from them, may be deemed reliable and trustworthy.

In future work, we would like to explore the conjecture that the similarity in blind and multimodal Encoder performance stems from the fact that the Guesser has access to the category of the target object. We would like to explore how the models would perform when they have to rely on visual information rather than textual object category.

Acknowledgements

We would like to express our gratitude to the people who made this replication project possible. Firstly, we would like to thank Alberto Testoni for all the helpful communications concerning the code that we needed to run to replicate the experiments, as well as his co-authors, Claudio Greco and Raffaella Bernardi. We would also like to thank Ravi Shekhar for his comments on the GDSE repository. Secondly, we would like to thank the MLCog Support team, and Philipp Sadler in particular, for helping us with GPU management. Finally, we would like to thank Explosion for their willingness to provide GPU resources.

References

Harm De Vries, Florian Strub, Sarath Chandar, Olivier Pietquin, Hugo Larochelle, and Aaron Courville. 2017. Guesswhat?! visual object discovery through multi-modal dialogue. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5503–5512.

Claudio Greco, Alberto Testoni, and Raffaella Bernardi. 2020. Which turn do neural models exploit the most to solve guesswhat? diving into the dialogue history encoding in transformers and lstms. In *NL4AI@ AI*IA*, pages 29–43.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Aditya Mogadala, Marimuthu Kalimuthu, and Dietrich Klakow. 2021. Trends in integration of vision and language research: A survey of tasks, datasets, and methods. *Journal of Artificial Intelligence Research*, 71:1183–1317.

Chinnadhurai Sankar, Sandeep Subramanian, Christopher Pal, Sarath Chandar, and Yoshua Bengio. 2019. Do neural dialog systems use the conversation history effectively? an empirical study. *arXiv preprint arXiv:1906.01603*.

Hao Tan and Mohit Bansal. 2019. Lxmert: Learning cross-modality encoder representations from transformers. *arXiv preprint arXiv:1908.07490*.

A Supplementary Graphs

This appendix includes the plots of the training progress of the models obtained with Weights & Biases (`wandb`).

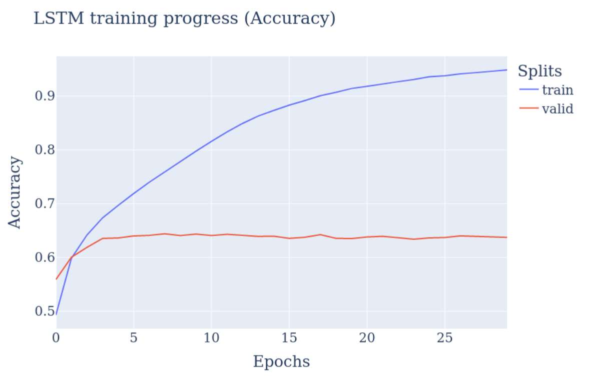

Figure 1: The training and validation accuracy of the LSTM Encoder model. We obtained the best validation accuracy on epoch 8, as compared to 19 reported in the original paper.

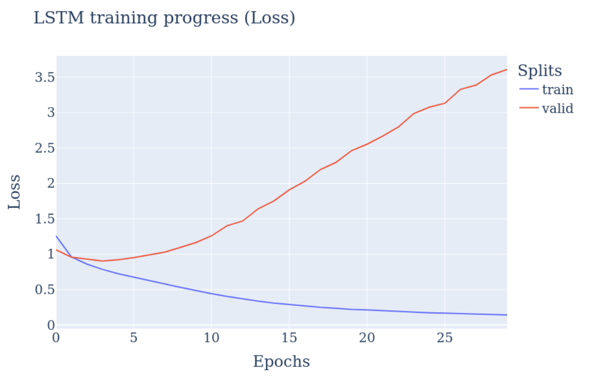

Figure 2: The training and validation loss of the LSTM Encoder model.

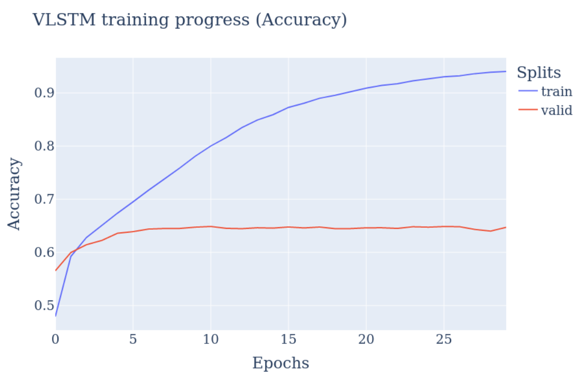

Figure 3: The training and validation accuracy of the V-LSTM Encoder model. We obtained the best validation accuracy on epoch 11, as compared to 9 reported in the original paper.

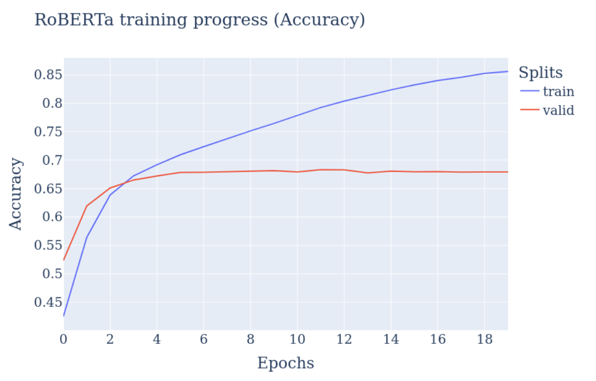

Figure 5: The training and validation accuracy of the RoBERTa Encoder model. We obtained the best validation accuracy on epoch 13, as compared to 7 reported in the original paper.

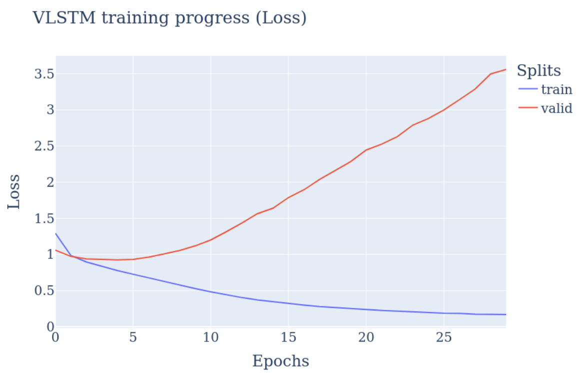

Figure 4: The training and validation loss of the V-LSTM Encoder model.

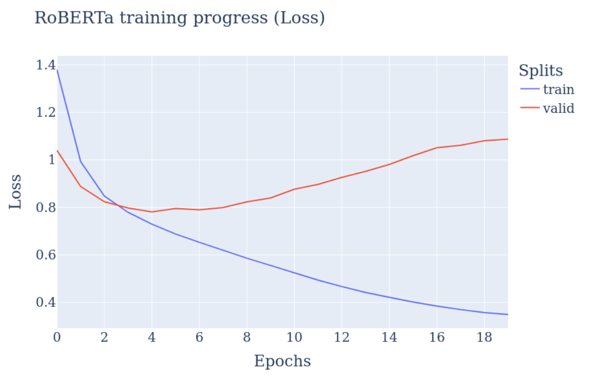

Figure 6: The training and validation loss of the RoBERTa Encoder model.

B Appendix: Atanasoska Tamara

I would like to preface this appendix by saying the group worked effectively and cohesively, with a lot of frequent online meetings and open, consistent chat. We've collaborated on every part of the project and have made all the big decisions together.

We are the same group since the beginning of the PM. Early on, during the individual paper presentations, I had the luck to have one with a great reproducibility section, and I created a document where we can already place these ideas for the upcoming project. We had one rivaling idea, but we picked the paper I was presenting.

In the initial week of the project, I took on the task to organise the infrastructure/ experiment setup, by creating a repository and cloning the repos, creating an "experiment and notes hub" where we continuously document our progress, and setting up meetings/feedback channels. My colleagues did the initial exploration of the repositories we planned to use. In the following week, I paired with Galina to make the code for the original GuessWhat?! Repository run. At some point, we split the task, and I kept at the last code issues as well as preparing the PRs to contribute back to the repository(and contacting the authors/maintainers) and Galina focused on preparing scripts for the data manipulation required for the experiments. During this time Bhuvanesh was focusing on communication with the authors and obtaining the resources in order to be able to run the other repository. We were in touch and had similar issues when "fixing" the code from both repositories.

Once we were done with the original GuessWhat?!, we focused on obtaining results from the models from the main Aixia2021 repository, writing documentation, and starting to integrate all that into the report. I focused mostly on writing the extensive documentation, creating easy to setup environments, and checking that all runs as expected and it is reproducible. Since we already had most of the models running, Galina took over the task of making sense of the results in a more meaningful way, while Bhuvanesh and I contributed to the content. When it comes to running LXMERT, I spent a day in the office reproducing the environment and resources for it on my GPU machine at work. Bhuvanesh and I paired on this when we realized that it is the lack of computational resources that has been also crashing our server the days before when we tried to run LXMERT. We tried to solve it, but we decided it is best to leave it for the second part of the project.

I have also been a driver when it comes to the communication and preparing the presentations of the project results/plans, of course with a great contribution from both of the other team members.

C Appendix: Ryazanskaya Galina

My role in the project consisted of several separate tasks. Firstly, I contributed to the conceptualization of the work we had to do, selecting which tasks we had to replicate, and what the specific implications of this choice would be for the project work that needed to be done. Secondly, I contributed to the alterations needed to be done to fix the bugs in the (De Vries et al., 2017) repository, as I was most familiar with `tensorflow`. Thirdly, I was responsible for the creation of the datasets needed for the experiments. Lastly, I was in charge of processing, interpreting, and reporting the results.

As for my teammates, Tamara was responsible for the organizational and specific project management as well as documentation and repository management, and has contributed to work on both repositories, while Bhuvanesh was the person responsible for fixing and running the code in the Greco et al. (2020) repository, the practical replication part of the experiments.

In my opinion, this replication project closely reflects the "real-life" work that has to be done when replicating and building upon the scientific results of others. Meaning, there has to be a lot of bug-fixing, library version control and environment setting. Secondly, separation of work in the team is a very good opportunity for speeding up the replication process, but there has to be a lot of clear communication and synchronisation, specifying up to the exact commands the process of running the code. Finally, in my opinion, this project was made possible by the server support and by the helpful remarks and instructions provided by the authors.

D Appendix: Verma Bhuvanesh

The initial hurdle that we had was to get a codebase for the paper that we were working on. I contacted the authors and one of the authors of the paper Alberto Testoni responded. We asked for a Github repo of the project and then various files that were missing. Once we had the repo, we faced another issue while running the code due to incompatible CUDA versions at the GPU server. I was in constant contact with Philipp Sadler (Support, MLCog) to handle this issue. With the server ready to use, Tamara added the project to the server. I started working on a newer repo and tried to make the training pipeline run but in the process did not go through the code and files available in depth. Later I realised that the newer repo contains all files that are required to completely reproduce the paper which led us to start working on the new repository altogether. We managed to fix minor issues in the train pipeline (most of them were similar which were already fixed in the original repo) and I integrated W&B to the script. Simultaneously, Tamara was documenting everything and Galina was adding all the information provided by us to the report. For testing, Galina created test files for the experiment. Meanwhile, me and Tamara tried our best to make the LXMERT model work but it did not work out and we were running out of time so we all decided to skip it for the reproduction part.