Replication / Natural Language Processing

ReDynamic Routing Transformer Network for Multimodal Sarcasm Detection

Andrea Infantino[1] and Eleonora Quaranta[1]
[1]University of Illinois at Chicago, Chicago, IL

A replication of tian-etal-2023-dynamic.

Reproducibility Summary

Scope of Reproducibility — This paper introduces a new architecture for sarcasm detection leveraging multimodal information from text and images. The authors claim that their model outperforms the state-of-the-art models on the Multimodal Sarcasm Detection dataset. With this work, we aim to reproduce these results and confirm or deny the authors' claims.

Methodology — Our reproducibility study was performed starting from the code provided by the authors, with minimal changes. We used the same dataset, evaluation metrics and model hyperparameters as the original paper. We used a Colab notebook to run the experiments on an NVIDIA V100 GPU.

Results — We were able to confirm the results reported in the paper to within 1.4% F1 score and 0.1% accuracy, therefore supporting the claim that the proposed model outperforms the state-of-the-art models on the Multimodal Sarcasm Detection dataset.

What was difficult — The setup of the experiment required some adjustments concerning the version of the required libraries. We also needed to fix some minor bugs in the code and perform some preprocessing operations locally.

What was easy — After the setup of the experiment was completed, running the code and reproducing the results stated in the paper was pretty straightforward.

Communication with original authors — No communication with the original authors was necessary.

Table 1. MSD dataset composition and split information

	Train	Development	Test
Sarcastic	8,642	959	959
Non-sarcastic	11,174	1,451	1,450
Total	19,816	2,410	2,409

1 Introduction

In this work we attempt the reproduction of the results presented in the paper ”Dynamic Routing Transformer Network for Multimodal Sarcasm Detection” [1], which proposes DynRT, a new architecture to tackle the sarcasm detection task. The authors of the paper leverage multimodal NLP to join information obtained from pictures and their corresponding captions.

2 Scope of reproducibility

The authors of the paper claim that their model outperforms the state-of-the-art models in the sarcasm detection task. They also claim that their model is able to leverage multimodal information to improve the overall performance in terms of both accuracy ad F1 score. The novel DynRT architecture is used to capture incongruity between the picture and the caption, which is a key aspect of sarcasm, and is tested on a multimodal sarcasm detection dataset to compare the results with other state-of-the-art models.

3 Methodology

To reproduce the results stated in the original paper we mainly used the code made available by the authors in their public repository (https://github.com/TIAN-viola/DynRT). The instructions provided were partially incomplete, and the code needed some small adjustments. We used a Colab notebook to run the experiments, which allowed us to leverage the NVIDIA V100 GPU.

3.1 Model description

The DynRT model is composed of three main parts: two encoders and a dynamic routing transformer. The two encoders are used to encode the input text and the image features, and are respectively RoBERTa [2] for the text and ViT [3] for the images. The dynamic routing transformer is used to capture the incongruity between the two modalities. It is composed of a stack of N layers, each of which is composed of a multi-head co-attention routing module, a multi-head self-attention module and a feed-forward network; each module is followed by a residual connection and a normalization layer. The transformer leverages hierarchical co-attention to module the incongruity between texts and images. The output of the encoders is fed to the dynamic routing transformer, which outputs a vector of *routed features* that are used for the classification together with the *image features*. The output of the classification is a vector containing the probabilities of the input being sarcastic or not.

3.2 Datasets

The dataset used to evaluate the architecture is the Multimodal Sarcasm Detection (MSD) dataset, whose composition and split information are shown in table 1. The dataset is not perfectly balanced, but it reflects the real-world distribution of sarcastic and non-sarcastic sentences. The dataset is composed of sentences and their corresponding images. The preprocessing operations performed on the dataset before feeding it to the model are the following:

- Discard tweets with regular worlds, such as *sarcasm, sarcastic, irony, jokes, humor, reposting, ironic, exgag,* and URLs
- replace mentions with the token <*USER*>

The dataset can be downloaded from https://github.com/headacheboy/data-of-multimodal-sarcasm-detection.

3.3 Hyperparameters

The configuration used to run the experiments is pretty similar to the one used by the authors (i.e. the hyperparameters that could have influenced the performance of the model, such as the learning rate, the batch size, etc., were left unchanged) in order to perform a fair comparison of the results. The only difference in our runs was in the parallelization of the training process, which was not configured by default. Initially we tried without it, but the training process was excessively long. Therefore, we increased the number of workers to 10. This has been made possible by the computational power and capacity provided by the GPU that we used. In this way we could run the whole training process (15 epochs) in less than 4 hours.

3.4 Experimental setup and code

Since we used a Colab notebook to run the experiments, we created a copy of our repository in a Google Drive folder, and we mounted the drive in the notebook. Then we tried installing the requirements in the way described in the readme.md of the original repository (https://github.com/TIAN-viola/DynRT) but we encountered some problems due to incompatible versions of some libraries. Therefore, we manually changed the installation process by updating the problematic libraries to their latest version. We then proceeded to the preprocessing phase as shown by the authors, which consisted in performing the cleaning operations on the dataset and in converting the images to tensor. This last operation was performed locally, since we encountered some issues when trying to run it on the Colab notebook. We then uploaded the preprocessed dataset to the notebook and we started the training process. Here, we encountered other issues. Firstly, we had no feedback about how the training process was proceeding, therefore we decided to increase the verbosity by adding a code line that printed the advancement of the number of batches processed (with the default batch size of 32 we had to elaborate 611 batches for each epoch). Secondly, we got an exception due to the loading of some .npy files containing the tensors (previously created starting from the images in the preprocessing phase). Specificaly, some files couldn't be serialized in the way the algorithm was originally written. To solve this, we added an exception handling code block to manage these files. Finally, since Colab's stability is not complety reliable, we added a simple script in our notebook that leverages the checkpoint system implemented by the original authors: it checks whether a checkpoint file already exists or not, and if it does it loads the model from it (in case of multiple checkpoints models, it load the last one). After these changes, we were able to run the training process without any other issues.

3.5 Computational requirements

As already mentioned, we used the NVIDIA V100 GPU provided by Colab Pro to run the experiments. The whole training and evaluation process took about 4 hours to com-

Table 2. Results obtained by the original authors and by us

	Original	Reproduced
Accuracy	93.49	93.59
F1	93.21	91.93

plete, and we experienced a GPU memory consumption of about 4 GB. It is interesting to note how with no parallelization a single training epoch took about 3 hours to complete, which would have led to an excessively long training process of 45 hours for 15 epochs. This would have been an issue since Colab only allows for 12 hours of continuous execution.

4 Results

4.1 Results reproducing original paper

The results obtained are in line with the ones provided in the paper. The authors report accuracy and F1 metrics obtained by averaging the performance of five different runs, showing how the newly introducted architecture outperforms other state-of-the-art models. The results that we obtained, shown in table 2 confirm those claims for accuracy results, but the F1 score obtained by us is slightly lower.

5 Discussion

Our experiment results match the ones reported in the paper, with a slight difference in the F1 score. This could be due to the fact that the authors averaged the results of five different runs, while we only performed one run. This could have led to a slightly different initialization of the model's weights, which could have influenced the final results. Anyway, the difference is not so significant, also because the F1 score obtained by us is still higher than the ones obtained by the other state-of-the-art models. One interesting thing to note was how the performance of the model increased during the training process. As we can see from figure (TODO: plot metrics), the accuracy of the model was growing significantly slowlier during the first epochs, and then had a peak around the (TODO: n)th epoch. Our interpretation of this phenomenon is that the low learning rate used (1e-6) caused the model to get stuck in a local minimum, and then when it was able to get out of it, it started to learn faster, as shown by figure (TODO: loss plot).

5.1 What was difficult

The setup of the environment was not straightforward, since the instructions provided by the authors were outdated. We had to manually change the installation process of the requirements, and we had to add some code lines to the training process to make it work. This has been even more challenging because of the lack of comments in the code. We also had to perform the preprocessing phase locally, since we encountered some issues when trying to run it on the Colab notebook. Also, it would have been interesting to compare the results in terms of precision and recall as well. Unfortunately this hasn't been possible since the authors didn't provide those metrics in their paper.

5.2 What was easy

After having solved the issues concerning the setup of the environment, the training process was pretty straightforward. We only had to run the notebook and wait for the process to reach an end. Moreover, we managed to replicate the results stated in the paper directly using the configuration provided by the original authors.

5.3 Communication with original authors

Since we managed to reproduce the results stated in the paper, we didn't have to contact the original authors.

References

1. Y. Tian, N. Xu, R. Zhang, and W. Mao. "Dynamic Routing Transformer Network for Multimodal Sarcasm Detection." In: **Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**. Ed. by A. Rogers, J. Boyd-Graber, and N. Okazaki. Toronto, Canada: Association for Computational Linguistics, July 2023, pp. 2468–2480.
2. Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. "RoBERTa: A Robustly Optimized BERT Pretraining Approach." In: **CoRR** abs/1907.11692 (2019). arXiv: 1907.11692.
3. A. Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale." In: **CoRR** abs/2010.11929 (2020). arXiv: 2010.11929.