Leveraging Transformers for Misinformation Detection in Multimodal Tweets

TIM NIKOLAAS LINDEIJER, Univerity of Stavanger, Norway
TORD MARTIN YTREDAL, Univerity of Stavanger, Norway
HENRIK VATNDAL, Univerity of Stavanger, Norway

Due to the widespread access to the internet and social media, misinformation is being spread to a wider audience. A study has shown that for a topic like Covid-19 which is often considered controversial, almost 25% of tweets contain some form of misinformation. Additionally, a vast amount of tweets are made each day making manually checking each one virtually impossible. In this project we aim to use automatically identify if a multimodal tweet consisting of text and an image needs to be fact-checked using transformers and LLMs.

Our results show that a fine tuned variant of the RoBERTa transformer, trained with the textual descriptions of the image along with the tweet text, is able to correctly label 85% of the tweets in our testing data. This model, which we call RoBERTa-LLaVA outperforms LLMs like LLaMA3 and inherently multimodal models like CLIP.

The GitHub repository for our project can be found here:
https://github.com/TimLindeijer/TiToHeDAT550.

ACM Reference Format:
Tim Nikolaas Lindeijer, Tord Martin Ytredal, and Henrik Vatndal. 2024. Leveraging Transformers for Misinformation Detection in Multimodal Tweets. 1, 1 (April 2024), 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

1.1 Background and Motivation

Since the invention of the internet, it has become easier for people to publish and consume unverified information. This is especially true for social media platforms as shown by a study on misinformation related to Covid-19 by [Kouzy et al. 2020]. In this study, 673 tweets related to trending hashtags on the topic were analyzed and 153 (24.8%) were found to contain misinformation. While this result is not representative for all content posted on the platform, it highlights how much misinformation is present in controversial topics such as Covid-19. Additionally, due to the high volume of new posts made these platforms, it is very costly to manually check each one for misinformation. One possible solution to mitigate this cost, is to leverage machine learning technology like transformers and large language models (LLMs) in order to filter out tweets that need to be fact-checked. Traditionally, such models has had a focus on the textual content on social media platforms which might not capture the full context of the post [Wang et al. 2021][Comito et al. 2023]. In this project we aim to tackle the *CLEF2023–CheckThat-Lab* Task 1a challenge, and use the provided dataset of multimodal content made up from both the text and image in a tweet in order to accurately detect if fact-checking is needed [Alam et al. 2023].

Authors' Contact Information: Tim Nikolaas Lindeijer, tn.lindeijer@stud.uis.no, Univerity of Stavanger, Stavanger, Norway; Tord Martin Ytredal, tm.ytredal@stud.uis.no, University of Stavanger, Stavanger, Norway; Henrik Vatndal, 27631@uis.no, University of Stavanger, Stavanger, Norway.

2024. ACM XXXX-XXXX/2024/4-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 BACKGROUND

2.1 Transformers

Transformers are deep neural networks first introduced in 2017 as a way to more efficiently perform tasks typically associated with recurrent neural networks (RNNs) [Vaswani et al. 2017]. RNNs maintain a sequence of hidden states related to previous words in a sentence, and uses this to process the next word. This makes RNNs slow to train especially for longer sequences as this process can not be done in parallel. Transformers use attention mechanisms which lets the model associate the different inputs when producing the output. This makes transformers more computationally efficient and better at capturing longer dependencies in sentences compared to RNNs [Vaswani et al. 2017][Singh and Mahmood 2021].

Transformers has been applied to many different tasks consisting of single or multimodal content such as images, text, speech or video. Because of the diversity of tasks these models can handle, variations of the original have been created. These variations can be split into three main categories. The first transformer proposed is an **encoder-decoder** transformer and tackles the task of machine translation [Vaswani et al. 2017]. Other notable encoder-decoder transformers include BART and T5 which can be used for machine translation, Q&A or semantic equivalence [Singh and Mahmood 2021]. **Encoder only** transformers are useful for tasks where a deeper understanding of context of the input is required, but where there is little use for a sequential output [Singh and Mahmood 2021]. An example of such a transformer is the Bidirectional Encoder Representations from Transformers (BERT) model which can be used in tasks like text classification and sentiment analysis. Lastly, **decoder only** transformers are good at generation sequences of text [Singh and Mahmood 2021]. The most well known decoder only transformers is Generative Pretrained Transformer (GPT) which has seen great popularity since OpenAI released ChatGPT to the public. In decoder only transformers the input is translated directly into tokens without the use of an encoder. The embedded tokens are then used to generate more tokens one by one [OpenAI et al. 2024]. This makes decoder only transformers well suited for tasks such as text completion and generation.

2.1.1 Attention Mechanisms. Attention can be thought of as the component which separates transformers from other neural networks.Self-attention allows a transformer to create dependencies between input words by associating them with each other. This allows the transformer to focus on specific characteristics of the input. By using multiple self-attention layers in parallel, more representations from different positions in the subspace can be extracted. This is known as multi-headed attention [Vaswani et al. 2017]. This works well especially in the encoder block of a transformer where the full context of the input sentence is needed. In the decoder however, having the full context is problematic as words/tokens should be generated only with the knowledge of previous tokens. To ensure that this auto-regressive property is maintained, masked multi-head attention is used in the decoder [Vaswani et al. 2017]. This sets the connections to tokens on the right to $-\infty$ effectively ignoring them.

2.2 Transformer-Based models

2.2.1 Robustly Optimized BERT Approach (RoBERTa). RoBERTa is an auto-encoded encoder only transformer based on the BERT transformer designed for text input. The main difference between these transformers comes from RoBERTa being trained on more data, for longer and in bigger batches compared to BERT [Liu et al. 2019]. Additionally, RoBERTa is trained without next sentence prediction loss, a binary classification loss meant to learn the model to recognize sequences belonging to the same document in for improved performance in natural language inference tasks. RoBERTa is also trained on longer sequences and applies dynamic masking during training [Liu et al. 2019]. These modifications allow RoBERTa to outperform BERT in most tasks. RoBERTa has also seen success in tasks like fact verification, where statements are evaluated to three categories depending on whether there is evidence to support the claim, refute the claim or if there is not enough information on the topic [Naseer et al. 2021]. This task is similar to the one tackled in this project, which suggests that RoBERTa will also work well for our task.

2.2.2 Contrastive Language–Image Pretraining (CLIP). CLIP is a multimodal transformer developed by OpenAI to perform zero shot classification tasks by connecting visual components with text. CLIP uses two separate transformer encoders for the input, for textual data it uses a standard transformer with multi-head attention and a visual transformer (ViT) for the image [Radford et al. 2021]. In a ViT, the image is tokenized by splitting it into smaller parts and converted to vectors. The rest of the structure is similar to other transformers as it also uses multi-head attention. CLIP is trained using contrastive loss, where the output from the two encoders is compared such that the embeddings for images and text describing similar concepts are pulled together, and different concepts are pushed away [Radford et al. 2021]. This allows CLIP to find connections between text and images. In the original paper, CLIP is used to predict which label from a set of potential labels belongs to an image. This means that CLIP better suited for tasks like classification. However, due to its multimodality, good zero shot performance and use in previous fact checking related research it might still work well for our task [Yao et al. 2023].

2.3 Large Language Models (LLMs) and Ollama

LLMs are generally transformer models which are able to generate text, understand context and hold conversations similar to how a human would. Compared to other transformers, they are trained on bigger and more diverse datasets while also containing more trainable parameters. This allows LLMs to perform well in a wide array of tasks even without precise fine tuning. Instead, simply giving context though prompts or the system message in a few-shot manner is often sufficient to guide these models to solve specific tasks [Minaee et al. 2024] [Zhao et al. 2023]. Ollama is an application made for running and managing of LLMs locally. Ollama supports several different models, both single- and multimodal with different sizes and parameter counts [Contributors 2024]. It allows for modifying parameters like temperature and system message in order to tailor the LLM's responses to specific tasks or use cases.

2.3.1 Zero-, one- and multi-shot prompting. Zero-, one- and multi-shot prompting refers to how much context is given to an LLM through examples before asking it to make predictions. This can be done through initial prompts or the system message. As the names suggest, zero-shot prompting gives no examples, one-shot gives one example and multi-shot gives several examples. Usually multi-shot prompting leads to the best results as the LLM is able to understand its task better given multiple examples.

2.4 LLaMA

Large Language Model Meta AI (LLaMA), is a series of transformer-based LLMs developed by Meta Platforms capable of handling complex language processing tasks. As of 2023 LLaMA models encompasses a collection of models ranging from 7B to 65B parameters, all of which are open source [Hugo et al. 2023].

2.4.1 LLaVA. Large Language and Vision Assistant (LLaVA), developed in 2023, is an end-to-end trained LLM capable of understanding both visual and texual data. The model achieves this by including a vision encoder, combined with the Vicuna LLM which provides general-purpose language understanding. This combination enables the model to complete tasks such as describing images, as well as providing a response to more ambiguous prompts such as "What is unusual about this image" or "What item does not belong" [Liu et al. 2023].

2.4.2 LlaMA3. LLaMA3 is Meta's newest entry in the LLaMA series consisting of two models of different sizes, with 8B and 70B parameters respectively. Boasting state-of-the-art (SOTA) performance across several tasks, both models are expected to place high on the leaderboards of within their classes, but are as of now only evaluated by Meta themselves (who placed them among top 5 whithin MMLU, GPQA, HumanEval, GSM-K and MATH) [Huang et al. 2024].

2.5 Fine-tuning

The models mentioned in this chapter are all pre-trained, meaning that they have already undergone training on large datasets which serve as a good starting point for the tasks they aim to tackle. By utilizing the knowledge gained from this previous training, fine-tuning allows for creating task specific models without large dataset at a reduced computational cost [Bergmann 2024].

2.6 Metrics

To quantify how well a model is performing in a binary classification task, it is necessary to have metrics to evaluate it on. These metrics can be calculated using the number of true positives, true negatives and total predictions made by the model. **Accuracy** is the measure of how many predictions were correct, out of the total number of predictions. This value ranges from zero to one where a higher score means that the model is making more correct predictions. **Precision** measures how precise the model is when it comes to predicting true positives. It calculates the ration between true positives to total positive predictions. A high precision means that when the model predicts that there is a positive, it is likely the correct label. This makes precision a good metric in a scenario where false positives are more harmful than false negatives. However, if the goal is to catch all positives and false positives are not much of an issue, **recall** might be a more useful metric. Recall measures the ratio of true positives to to all positives, giving a measure of how well the model can identify positive samples. **F1 score** combines precision and recall by taking the harmonic mean of both values. By taking this mean, the f1 score finds a balance between precision and recall which makes it a good metric for measuring how well a model minimizes both false positives and false negatives.

2.7 Bootstrapping

In statistics, the bootstrap method is a way to calculate the standard deviation or error for a given statistic such as the mean or median [Johnson 2001]. This concept can be used when testing a model to get an accurate prediction for how it might perform on real data. This is achieved by repeatedly sampling the testing dataset with replacement, making predictions and calculating the metrics for each sample. Then the mean and standard deviation or error can be calculated for each metric. Allowing repetition of entities in samples is a key part of bootstrapping, because it better allows for capturing variability in the testing data [Michelucci and Venturini 2021]. Without replacement, each bootstrap sample would simply be a random subset of the original data.

3 METHOD AND DATA

3.1 Data

The dataset used for training and testing the models we used is the English data for task 1A in the CLEF2023–CheckThat-Lab [Alam et al. 2023]. This dataset contains the text and images of tweets and is already split into three categories meant for training, validation and testing. The dataset contains a total of 3174 tweets and accompanying images.

3.1.1 Pre-processing. Before training we applied some pre-processing to our data. For CLIP, we removed URLs, special characters, newlines, multiple spaces in addition to leading and trailing spaces. For RoBERTa and the Ollama models, no particular text cleaning was applied except for fetching the relevant parts of the data such as tweet id, the text and label. Lastly, for the Ollama models the images were base64 encoded so they fit with the expected datatype for the LLaVA model.

3.2 Setup

The programming language used throughout this project is Python 3.11 [Team 2022]. To build, train and test our models, we use the open source machine learning library pytorch [Paszke et al. 2019]. Additionally, we use Ollama to run the LLMs [Contributors 2024]. The models were trained on the UiS Unix servers using the GPU's under the slurm system on Gorina11. The Ollama models were run locally on our personal computers.

3.2.1 Hyperparameters. When training of our transformers and initialization of the LLMs, we use different hyperparameters depending on the model and their requirements. For RoBERTa and CLIP, RoBERTa seems to converge faster compared to CLIP thus needing fewer epochs and lower learning rate when fine tuning. For CLIP we use a learning rate of 1e-3, batch size of 32 and run for 90 epochs. A learning rate of 1e-5, batch size of 16 and 10 epochs was used for RoBERTa. Additionally the loss used is different. For RoBERTa we used the default loss in the pytorch trainer class which is cross entropy loss. When training CLIP, we created our own training method as the trainer did not work properly, this also allowed us to select the type of loss which we set to binary cross entropy with logits (BCEWithLogitsLoss in pythorch). Both models use Adam as the optimizer.

When tuning our LLMs, we set the temperature to 0.1 in order to keep them more factual and stop them from "hallucinating" which is when the LLM invents information in its responses. We also tried several different system messages as described bellow.

(1) Answer yes if the prompt contains a highly debated subject or is harmful, no if else. Only single word response
(2) Answer yes if the prompt contains subjects that require additional fact checking, no if else. Only single word response
(3) Answer yes if the prompt contains controversial subjects, no if else. Only single word response
(4) You are a moderate on twitter tasked with labelling tweets for additional fact checking. Answer yes if the prompt contains misleading information, controversial themes or harmful content, no if else. Only single word response
(5) Answer yes if the prompt contains misleading information, no if else. Only single word response
(6) Answer yes if the prompt contains a verifiable factual claim, no if else. Only single word response
(7) Answer yes if the prompt contains harmful content, no if else. Only single word response
(8) Answer yes if the prompt contains harmful content or controversial subjects, no if else. Only single word response
(9) You are an assistant that reviews user input for claims that may need fact-checking. Your role is to identify statements that seem factual but could potentially be inaccurate, misleading or controversial. You should not attempt to fact-check the claims yourself, but rather flag any content that might need further investigation by a human fact-checker. Answer yes if the claim needs to be fact-checked and no if it does not need to be fact-checked. Only answer with yes or no.

3.2.2 Metrics. We use the same metrics explained in the previous chapter, namely accuracy, precision, recall and f1 score to evaluate the performance of our models and LLMs. Overall, accuracy and recall are the most important metrics. But the other scores are also taken into consideration. Due to our goal of detecting if a tweet needs fact checking, recall is a good measure to see how well the models can correctly classify positive instances. Looking at both recall and accuracy, we can get a good estimate on how well the model performs overall while focusing on detecting as many positives as possible.

3.3 CLIP

When experimenting with CLIP, we tried two main configurations. The base model pulled directly from Huggingface and a fine tuned model trained with the hyperparameters listed earlier in the chapter. Comparing these models will allow us to see the effects of fine tuning CLIP to our task.

3.4 RoBERTa

For RoBERTa, we also tried the base model and a fine tuned version. In addition to this, we also tried a fine tuned version trained with textual descriptions of the attached image. This is because RoBERTa only supports text an not images as input. We used LLaVA to describe all of the images and concatenated this with the accompanying tweet's text. With these models, we can examine the impact of fine tuning RoBERTa and how adding multimodality affects the outcome.

3.5 LLaMA3 and LLaVA

As mentioned above, LLaVA was only used to translate the images into textual descriptions. In addition to being used in RoBERTa, these descriptions were included when making predictions with LLaMA3, the LLM of our choice for this project. When it comes to experimenting with the LLaMA3 model, we tried a few diffrent approaches with varying success. First, we tried the different system messages and calculated the accuracy of the model in a zero-shot fashion. We then picked the one which yielded the best result which we used throughout the other experiments.

Next, we attempted to fine tune the LLaMA3 model. However, because of how Ollama runs on localhost, we were unable to use the UiS Unix system for this tuning. This is a problem because our machines are not powerful enough to train models of this size. Instead, we used Autotrain, a tool from Huggingface along with a free Google colab notebook. Unfortunately, the resources available with a free account, especially GPU memory proved to be an issue. To circumvent this, we attempted smaller models such as Mistral (7B parameters), Phi (3,8B) and Gemma (2B), but were unable to make them work with the whole training dataset. In a last ditch attempt, we used sharded 8B Llama-2-bf16 model and only 100 rows of data without image descriptions. This trained successfully, but was quickly discovered to be hallucination prone and ignoring system messages during inference. Due to this, we abandoned the effort to fine tune the LLaMA model.

Instead, we experimented with few-shot prediction using LLaMA3. In addition to the system message we provided the model with six examples and the accompanying label. This allows us to see how zero-shot and few-shot affects the models performance.

3.6 Bootsrapping

To get the metrics for our results, we use bootstrapping where we sample the entire test dataset of 548 tweets 100 times. We did this for all the models in order to get an estimate for how they handle unseen data.

4 RESULTS

In this section, we present the results of the CLIP, RoBERTa, and LLaMA3 models. Firstly, we will examine how training on the dataset influences the base models. Subsequently, we will compare the fine-tuned models with the base models, along with the disparity in results between the two fine-tuned models of CLIP and RoBERTa, as well as the outcomes of the LLaMA3 model.

4.1 Training

This subsection displays the graph gained from training the base CLIP and RoBERTa models.

4.1.1 CLIP. The graph 1, illustrates the progression of accuracy for the CLIP model with each step. In particular, there is an immediate increase from 67% to 75% at the start, followed by a gradual increase until it stabilises at 80% accuracy.

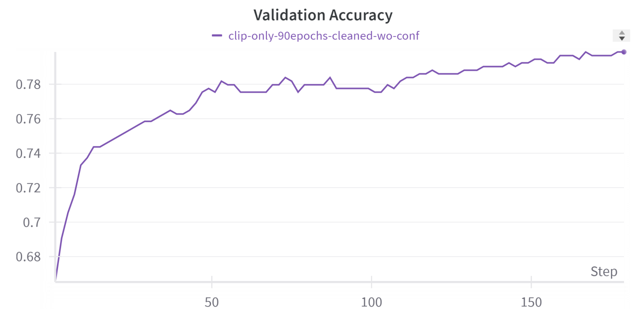

Fig. 1. Clip Validation Accuracy

The train loss in graph 2 shows how the loss get lowered the most at the start from 67% to 48%, and then a slow drop off towards 33% train loss.

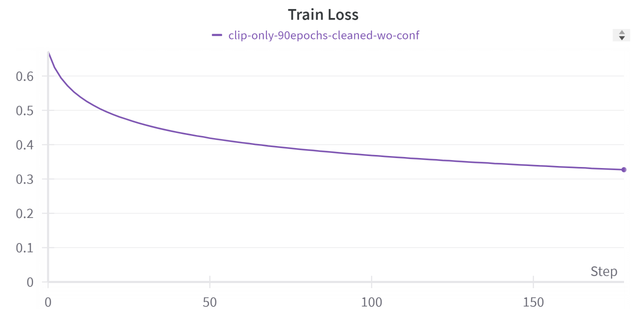

Fig. 2. Clip Train Loss

4.1.2 RoBERTa. Graph 3 shows the rapid increase in accuracy of the RoBERTa model to the data, with minimal subsequent increases afterwards. It starts at 83% accuracy and ends at 86%, indicating a plateau in improvement after initial adaptation.

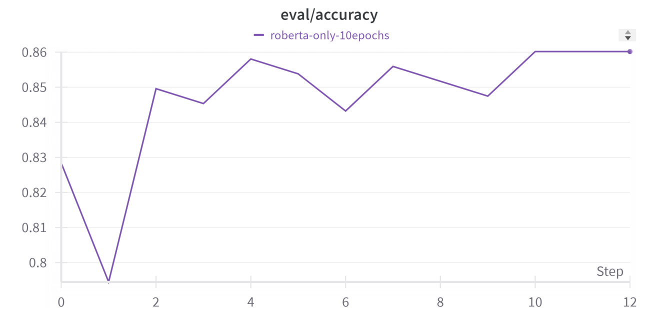

Fig. 3. Roberta Evaluation Accuracy

4.1.3 *CLIP fine-tuning comparison.*

Table 1 show the difference between a base CLIP model and a fine-tuned CLIP model. Here we see that the base model perform worse than the fine-tuned in accuracy and precision. The base model got 0.55 ± 0.020 for accuracy and 0.30 ± 0.033 for precision. The fine-tuned model got 0.67 ± 0.018 and 0.38 ± 0.077 these two metrics. On the other hand, the base model perform better in recall and F1, which scored 0.31 ± 0.035 and 0.30 ± 0.031 respectively. The fine-tuned model performed worse in the metrics and got 0.08 ± 0.020 in recall and 0.13 ± 0.030.

	Base	Fine-tuned
Accuracy	0.55 ± 0.020	0.67 ± 0.018
Precision	0.30 ± 0.033	0.38 ± 0.077
Recall	0.31 ± 0.035	0.08 ± 0.020
F1	0.30 ± 0.031	0.13 ± 0.030

Table 1. Comparison between base CLIP vs fine-tuned CLIP

4.1.4 *RoBERTa fine-tuning comparison.*

Table 2 shows how a base RoBERTa model, a fine-tuned model, and a fine-tuned model combined with image descriptions using LLaVA compare. The base model performs worse in all categories, while the fine-tuned model outperforms the base, and the fine-tuned model with LLaVA performs the best. The base model achieves an accuracy of 0.68 ± 0.019, whereas the fine-tuned model achieves 0.85 ± 0.014, and the fine-tuned model with LLaVA achieves 0.85 ± 0.016. There is a noticeable difference in precision, with the base model scoring 0.47 ± 0.026, compared to the fine-tuned model 0.75 ± 0.003, and the fine-tuned model with LLaVA 0.85 ± 0.015.

	Base	Fine-tuned	Fine-tuned w/ LLaVA
Accuracy	0.68 ± 0.019	0.85 ± 0.014	0.85 ± 0.016
Precision	0.47 ± 0.026	0.75 ± 0.003	0.85 ± 0.015
Recall	0.68 ± 0.019	0.81 ± 0.003	0.85 ± 0.016
F1	0.55 ± 0.025	0.78 ± 0.002	0.85 ± 0.015

Table 2. Comparison between base RoBERTa, fine-tuned RoBERTa, and fine-tuned RoBERTa with image descriptions created using LLaVA

4.1.5 *LLaMA3.*

3 shows a comparison between the zero-shot model and a few-shot model. The few-shot model included a model file with six rows of prompts containing the correct answer. Both models were LLaMA3 models, with the only difference being the extended model file of the few-shot model. These results were gathered by iterating over 100 samples 10 times. The zero-shot model scored higher in every metric compared to the few-shot model. It achieved 0.64±0.045 in accuracy compared to the few-shot model's 0.62 ± 0.046. It had a higher precision, obtaining 0.44 ± 0.080, while the few-shot model achieved 0.38 ± 0.086. The zero-shot model attained a recall score of 0.48 ± 0.096, compared to the few-shot model's 0.41 ± 0.092. Lastly, the zero-shot model performed better in F1 with 0.45 ± 0.077, whereas the few-shot model achieved 0.39 ± 0.078.

	Zero-shot	Few-shot
Accuracy	0.64 ± 0.045	0.62 ± 0.046
Precision	0.44 ± 0.080	0.38 ± 0.086
Recall	0.48 ± 0.096	0.41 ± 0.092
F1	0.45 ± 0.077	0.39 ± 0.078

Table 3. Comparison between Zero-shot and Few-shot LLaMA3 contro model

4.1.6 *Comparison of various models.*

Table 4 displays the differences between the fine-tuned models of CLIP, RoBERTa-LLaVA, and the local LLaMA3-LLaVA combination. The table shows that LLaMA3-LLaVA had the lowest accuracy at 0.64 ± 0.045, followed by CLIP with an accuracy result of 0.67 ± 0.018, and then RoBERTa-LLaVA at 0.85 ± 0.016. CLIP performs the worst in precision at 0.38 ± 0.077, followed by LLaMA3-LLaVA at 0.44 ± 0.080, and RoBERTa-LLaVA is significantly ahead at 0.85 ± 0.015. RoBERTa-LLaVA consistently maintains the lead in recall with a result of 0.85 ± 0.016, followed by LLaMA3-LLaVA at 0.48 ± 0.096, and CLIP lags far behind at 0.08 ± 0.020. The F1 results are quite similar, with RoBERTa leading at 0.85 ± 0.015, followed by LLaMA3-LLaVA at 0.45 ± 0.077, and CLIP far behind at 0.13 ± 0.030.

	CLIP	RoBERTa-LLaVA	LLaMA3-LLaVA
Accuracy	0.67 ± 0.018	0.85 ± 0.016	0.63 ± 0.052
Precision	0.38 ± 0.077	0.85 ± 0.015	0.43 ± 0.090
Recall	0.08 ± 0.020	0.85 ± 0.016	0.45 ± 0.094
F1	0.13 ± 0.030	0.85 ± 0.015	0.44 ± 0.083

Table 4. Results from bootstrap. The table compares the fine-tuned CLIP model, the RoBERTa model using image descriptions created by LLaVA, called RoBERTa-LLaVA, and the LLaMA models using image descriptions created by LLaVA, called LLaVA

4.2 Discussion

Looking at the results we can see that fine-tuning the base model can yield an improvement in performance. This can be seen in table 2 which compares the different RoBERTa models. Here we see a significant improvement in every metric. The fine-tuned model is more often correct, as well as having a lower deviation in results. Additionally, adding the image descriptions made by LLaVA, we see an improvement in precision, recall and F1. This demonstrates the benefits of multi modality over uni-modality.

At first glance, the same appears to be true for fine-tuning the CLIP model. However, when looking closer at the results in Table 1, we see that the recall is alarmingly low. This implies that the fine-tuned model has been trained to almost always answer no. This could be because of the way CLIP is designed as discussed in Section 2.2.2. Primarily meant as a multimodal classifier, CLIP is designed to establish and find connections between images and texts, not to derive abstract conclusions from them. This could be the reason why CLIP performs poorly, even when compared to the models that are not multimodal.

The comparison between zero-shot and few-shot LLaMA3 models revealed that the addition of examples created noise in our results. This noise could, for example, cause all prompts containing an arbitrary word like "Australia" to always be labeled "Yes" because the model mistakenly labeled a tweet containing the word as "Yes". Time constraints became a limiting factor when it came to the local models, as testing was quite time-consuming. The initial results served as the deciding factor in determining where to focus further efforts, which turned out to be the zero-shot model.

The findings from Table 4 reveal clear distinctions between the fine-tuned models of CLIP, RoBERTa-LLaVA, and the zero-shot LLaMA3-LLaVA. RoBERTa-LLaVA consistently performs better than the other models across various metrics, such as accuracy, precision, recall, and F1 score. Conversely, CLIP exhibits lower performance, especially in precision and recall. These results tell us that fine-tuning RoBERTa and providing it with description of the images connected to tweets is the best of the three models at deciding the check-worthiness of a tweet.

As for the fine tuning endeavour. Model behaviour is trained behaviour, as the training data was limited the desired behaviour/output was not present. A fix could be running our hugging face model locally in order to alter the behaviour with a model file. A pre-requisite would be that the model is able to understand instructions to a high enough degree which is not certain it would i.e have a concept of what controversial means. A guaranteed fix: Pay more, get more. With more hardware resources fine tuning would be both faster and provide more accurate results. This served as a good lesson/example in the limitations of freeware in a big data world.

5 CONCLUSION

This project aimed to explore if we could train models to predict if a tweet of multimodal content containing both text and an image, needs to be fact checked. We tested and compared models of different types and modality such as transformers and LLMs. Our findings suggest that the RoBERTa-LLaVA model, which is fine tuned with both text and image description from tweets performs the best at this task reaching an accuracy of 0.85 ± 0.016 and a precision of 0.85 ± 0.016. The RoBERTa-LLaVA model outperforms LLMs like LLaMA3-LLaVA as well as inherently multimodal models such as CLIP.

Looking at these results, our goal for this project were met and we are able to successfully identify if a multimodal tweet needs to be fact checked using our best performing model.

5.1 Distribution of work

When distributing our work, we decided early to split up the work where Tim was primarily responsible for running the transformer models such as CLIP and RoBERTa on the UiS slurm systems. Henrik was primarily responsible for the LLMs and the experiments conducted with them. Tord was primarily responsible for the report. Additionally, each member helped in coordinating experiments and writing. As responsible for the models, Tim and Henrik were the primary authors for the results section as well as using LLaVA to convert from image to text for use in RoBERTa-LLaVA. Tord helped with tasks such as designing experiments for the LLMs and transformers, which Henrik and Tim ran. Overall, each group members contributions added up to similar amounts and there were no issues with group dynamics.

REFERENCES

Firoj Alam, Alberto Barrón-Cedeño, Gullal S Cheema, Sherzod Hakimov, Maram Hasanain, Chengkai Li, Rubén Míguez, Hamdy Mubarak, Gautam Kishore Shahi, Wajdi Zaghouani, et al. 2023. Overview of the CLEF-2023 CheckThat! lab task 1 on check-worthiness in multimodal and multigenre content. *Working Notes of CLEF* (2023).

Dave Bergmann. 2024. Fine-tuning. https://www.ibm.com/topics/fine-tuning Accessed: 2024-04-23.

Carmela Comito, Luciano Caroprese, and Ester Zumpano. 2023. Multimodal fake news detection on social media: a survey of deep learning techniques. *Social Network Analysis and Mining* 13, 1 (2023), 101.

Ollama Contributors. 2024. Ollama. https://github.com/ollama/ollama Accessed: 2024-04-22.

Wei Huang, Xudong Ma, Haotong Qin, Xingyu Zheng, Chengtao Lv, Hong Chen, Jie Luo, Xiaojuan Qi, Xianglong Liu, and Michele Magno. 2024. How Good Are Low-bit Quantized LLaMA3 Models? An Empirical Study. arXiv:2404.14047 [cs.LG]

Touvron Hugo, Lavril Thibaut, Izacard Gautier, Marie-Anne Lachaux Xavier Martinet, Lacroix Timothee, Rozière Baptiste, Eric Hambro Naman Goyal, Azhar Faisal, Rodriguez Aurelien, and Joulin Armand. 2023. Open and Efficient Foundation Language Models. *Meta AI* (2023).

Roger W Johnson. 2001. An introduction to the bootstrap. *Teaching statistics* 23, 2 (2001), 49–54.

Ramez Kouzy, Joseph Abi Jaoude, Afif Kraitem, Molly B El Alam, Basil Karam, Elio Adib, Jabra Zarka, Cindy Traboulsi, Elie W Akl, and Khalil Baddour. 2020. Coronavirus goes viral: quantifying the COVID-19 misinformation epidemic on Twitter. *Cureus* 12, 3 (2020).

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual Instruction Tuning. In *NeurIPS*.

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:1907.11692 [cs.CL]

Umberto Michelucci and Francesca Venturini. 2021. Estimating neural network's performance with bootstrap: A tutorial. *Machine Learning and Knowledge Extraction* 3, 2 (2021), 357–373.

Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large Language Models: A Survey. arXiv:2402.06196 [cs.CL]

Muchammad Naseer, Muhamad Asvial, and Riri Fitri Sari. 2021. An Empirical Comparison of BERT, RoBERTa, and Electra for Fact Verification. In *2021 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*. 241–246. https://doi.org/10.1109/ICAIIC51459.2021.9415192

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel

Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems* 32 (2019), 8024–8035.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 8748–8763. https://proceedings.mlr.press/v139/radford21a.html

Sushant Singh and Ausif Mahmood. 2021. The NLP cookbook: modern recipes for transformer based deep learning architectures. *IEEE Access* 9 (2021), 68675–68702.

Python Core Team. 2022. *Python: A dynamic, open source programming language.* Python Software Foundation. https://www.python.org/ Python version 3.11.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

Zuhui Wang, Zhaozheng Yin, and Young Anna Argyris. 2021. Detecting Medical Misinformation on Social Media Using Multimodal Deep Learning. *IEEE Journal of Biomedical and Health Informatics* 25, 6 (2021), 2193–2203. https://doi.org/10.1109/JBHI.2020.3037027

Barry Menglong Yao, Aditya Shah, Lichao Sun, Jin-Hee Cho, and Lifu Huang. 2023. End-to-End Multimodal Fact-Checking and Explanation Generation: A Challenging Dataset and Models. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval* (<conf-loc>, <city>Taipei</city>, <country>Taiwan</country>, </conf-loc>) *(SIGIR '23)*. Association for Computing Machinery, New York, NY, USA, 2733–2743. https://doi.org/10.1145/3539618.3591879

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A Survey of Large Language Models. arXiv:2303.18223 [cs.CL]