Project title
This is Going Swimmingly!

Team members
Isabelle Shaprio (iashapir), Catherine Yu (cyu83), Jake Regenwetter (jregenwe), Aaron Martin (amart172)

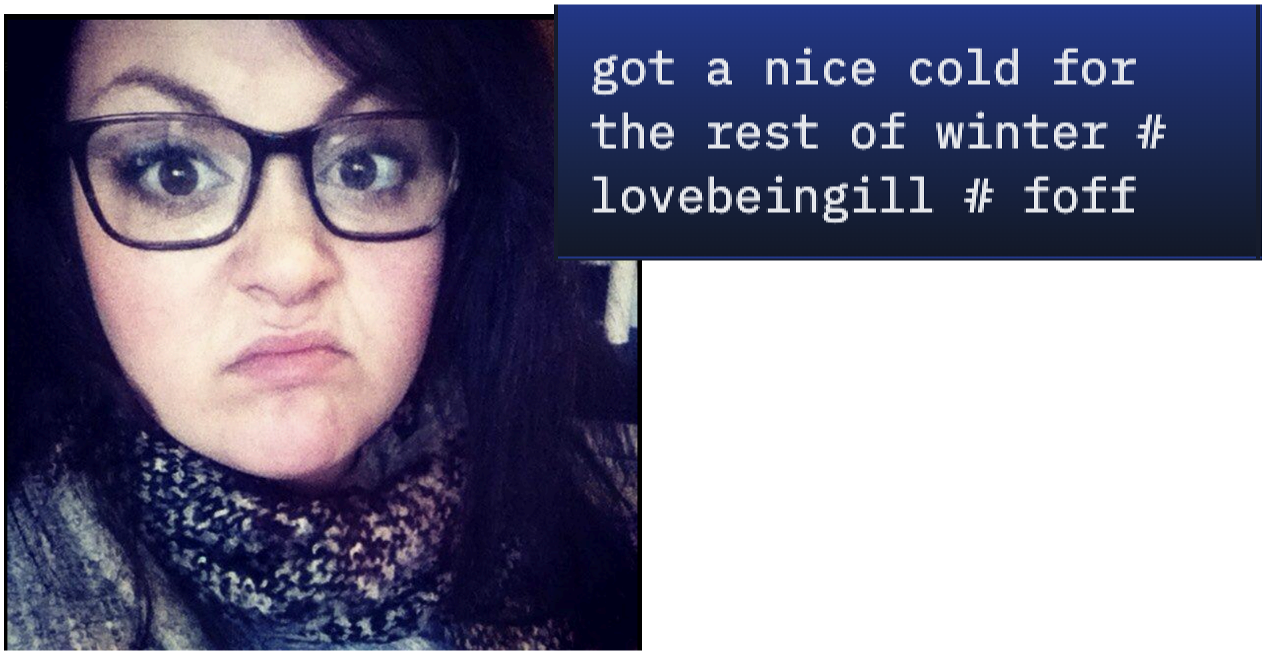
Introduction
Our project is based off of a paper implementing a relational context learning and multiplex fusion network to address the task of multimodal sarcasm detection - classifying several modes of inputs as sarcastic or not. This classification task is important because it can help filter misinformation online, and is also helpful for other machine learning tasks such as sentiment analysis. The paper focuses on implementing sarcasm detection *without* the use of graph structures, which is used by many existing sarcasm detection models. Using graph structures presents several limitations - most importantly, it is extremely computationally expensive to create graph networks modeling images and text. Instead, the model aims to understand dynamic relationships between the image and text pairs to capture the context needed to classify something as sarcastic or not. We chose this paper because it addresses a problem that we are all interested in and it builds off of concepts we've discussed previously in class - such as various types of attention.

Related work
The RCLMuFN paper goes into significant detail about several related works on both unimodal and multimodal sarcasm detection. It also compares the results of the RCLMuFN model with other successful unimodal and multimodal models' results. Thus, here, we will discuss work on this subject that has been done since the RCLMuFN paper was published. This paper, recently released to Arxiv by researchers at Tianjin University, Zhengzhou University, and The Hong Kong Polytechnic University, is in preprint and still under review. It introduces a new multimodal large-language sarcasm-detection model called Commander-GPT. This model diverges largely from other multimodal models similar to RCLMuFN, as it uses one module, the "commander" to oversee the sarcasm detection task and delegates intermediate steps to other specialized modules. The research team used GPT-4o as the "commander" and experimented with many other "civil" and "military officials" (modules) such as MiniCPM-V-2 and Claude-3-haiku to perform intermediate analysis tasks. The commander, upon receiving information from the "civil" and "military" officials, proceeds to perform sarcasm detection on the MMSD and MMSD 2.0 datasets using all the information it has gathered. Throughout the course of the detection, the commander is prompted using a prompting method, several of which were tested including Zero-shot CoT, Automatic Prompt Engineer, and Plan-and-Solve, to guide its course of action in selecting submodules.

Overall, the Commander-GPT model had significantly lower accuracy results on both datasets than RCLMuFN, with its highest accuracy being 74.5% on MMSD. That being said, its prompting and delegating design could prove to be much more generalizable to other tasks, data types, and networks in the future, thus it is still a considerable breakthrough in sarcasm detection research. If the strategies from both papers are combined, for example, if Commander-GPT used RCLMuFN as a submodule, an ever more powerful model could arise.

Data

In the original RCLMuFN paper, the model was trained on the MMSD and MMSD 2.0 datasets. The MMSD dataset contains pairs of images and related text sourced from social media platforms, some of which are sarcastic in nature, while others are not. Each pair is labeled as sarcastic or not by human annotators. For example, in one image, a woman with a grumpy expression is depicted, and the accompanying text, which superficially celebrates her falling ill, is clearly sarcastic.



got a nice cold for the rest of winter # lovebeingill # foff

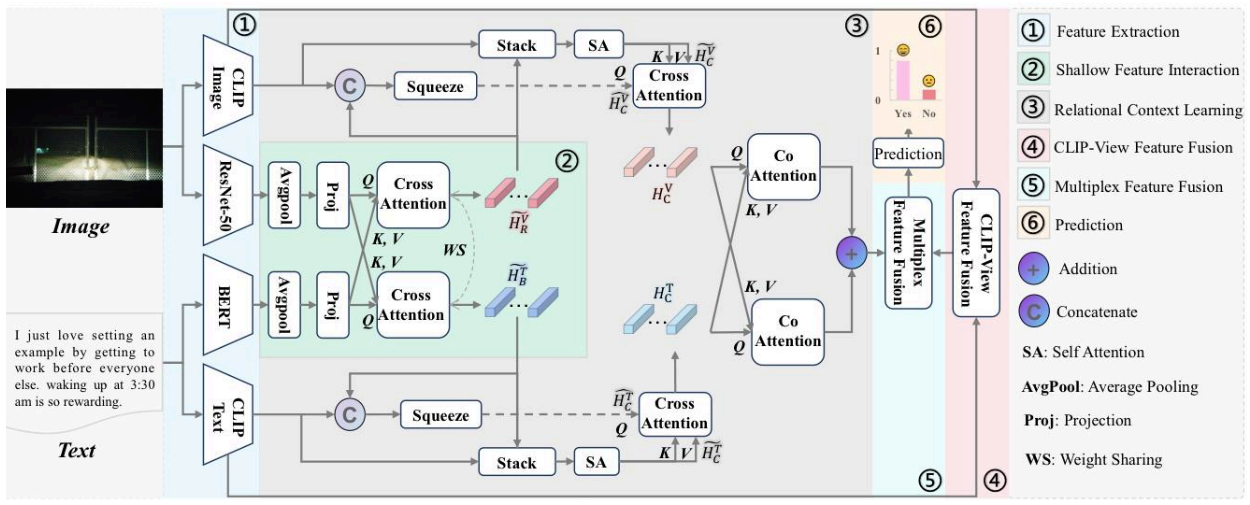
The MMSD 2.0 dataset differs from its precursor by removing problematic cues like hashtags and emojis, which were misleading models to train on superficial associated patterns rather than genuine indicators of sarcasm. It was also re-annotated to ensure higher quality and consistency. We intend to train our model exclusively on MMSD 2.0 to ensure the quality of our model and for simplicity's sake.

Specifically, the MMSD 2.0 dataset contains a total of 24,635 samples, already divided into 19,557 training, 2,387 validation, and 2,373 test samples. The text is broken down using the BERT tokenizer., which splits sentences into subword units. All tokenized sequences were padded or clipped to a maximum length of 77 tokens. Text features were encoded to 768-dimensional vectors using BERT. All images were resized to 224x224 pixels, and encoded using ResNet-50 to produce 768-dimensional embeddings for CLIP compatibility.

We also found a few additional datasets not used by the paper to test our model implementation on. The first is the SarcNet dataset, which contains image-text pairs that contain both English and Chinese captions. This would be interesting to test because the paper describes one limitation of the RCLMuFN model as only performing well on English text. We also found the MORE dataset, which only contains sarcastic image/text pairs.

Methodology

This diagram depicts the overall architecture of the RCLMuFN Model.



The model contains six main components, which we'll walk through here.

Feature Extraction (1)

Rather than write our own feature extractors, we will use pretrained models: CLIP Image and Text encoder, ResNet-50 and BERT. CLIP-Image and ResNet-50 will be used for image feature extraction, while CLIP-Text and BERT will be used for text-feature extraction. In this module, we will simply run the pretrained models on our input data.

Shallow Feature Interaction Module (2)

This module uses the feature extraction from ResNet-50 (images) and BERT (text), runs pooling and scaling operations on the features to standardize them, and the performs cross attention between the two sets of features to learn relational context. First, it runs Adaptive Average Pooling on the initial output features from the pretrained models, which supposedly reduces the model's dependence on specific features and makes the model more applicable to any random images and text. Next, we map the pooled image and text features to be the same dimensions for cross attention. For cross-attention, we use the **same** transformer layer for two operations: first with the image features as queries and second with the text features as queries. This transformer layer is a class of our own creation, TransformerCrossLayer, which contains linear layers, dropout, and normalization. The result of these operations is a set of image features that pays attention to text features and a set of text features that pays attention to image features.

Relational Context Learning Module (3)

The goal of this module is to first learn the context of the image and text features separately, then fuse these features. First, we will concatenate the outputs from step (2) with the CLIP feature extraction results from step (1) that we haven't yet used. The CLIP-Image features are concatenated with the step (2) image features and the CLIP-Text features follow suit. Next, we perform self attention on the text and image features respectively using TransformerEncoderLayers, a class we will write. This enhances our model's understanding of what the text and image mean separately. Finally, we perform cross-attention on the self-attention outputs (once we ensure that dimensions match up). We use two more TransformerCrossLayers to do this, with the post-self-attention image features as queries and the post-self-attention text features as keys and values and vice-versa. Finally, we perform a weighted sum of the outputs of these two cross-attention layers (0.6 for images and 0.4 for text was the RCLMuFN researchers determined was optimal).

CLIP-View Feature Fusion Module (4)

Since the CLIP pretrained module is so good at feature extraction, we end up putting a lot of weight in the information it provides us with. This module fuses the raw image and text features from the pretrained modules to ultimately create features that we weigh with the output of step (3) to assist in prediction. For the feature-fusion part, we perform cross attention using the exact same cross attention layer from the previous module in a similar way: first with images as queries, second with text as queries. We then created a weighted sum (similar to the above module), but this time, the text features are weighted more the the image features.

Multiplex Feature Fusion Module (5)

Our final step before predicting the class of our text/image pair is to fuse the two streams of cross-attention that we've created from the previous two modules. Naturally, we will do this in a non-obvious and relatively convoluted way. We first concatenate the weighted outputs of steps (4) and (3) and pass these features through a small MLP consisting of a linear layer, ReLU, and Sigmoid. We separately pass the same concatenated features through a similar MLP that uses Layer Norm instead of Sigmoid. The outputs of these two MLPs are multiplied, then weighted with H^{CLIP}, the output of step (4) (illustrating the importance of the CLIP feature extractor in our overall model). The RCLMuFN model uses a 50/50 weighting on these two outputs.

Prediction Module (6)

Finally, we pass the output of step (5) into a simple linear layer and apply softmax to get the output probabilities of each class (sarcastic or not).

Metrics

Our model seeks to solve a classification problem with just two classes: sarcastic and not. Thus, we can easily measure our model's success using the same metrics the paper uses. These are accuracy, precision, recall, and a micro-average F1 score. The paper conducts thorough experiments to collect metrics on the RCLMuFN model through both comparison experiments, an ablation study, and parameter analysis. These experiments seek to prove the success of their model compared to other models and validate their design decisions.

The comparison experiments compared their model's metrics with other multimodal as well as unimodal models trained on the MMSD and MMSD 2.0 datasets. We will compare our own model's metrics to those presented in the study to judge our accuracy against the original RCLMuFN model and other classifiers. The researchers also conducted an ablation study that tested the significance of each component of the model. To do this, they removed different parts of the model, ran it, and compared its metrics to the original. Through this, they determined that the most significant modules were the CLIP-View Feature Fusion Module and Shallow-Feature Interaction Module. We don't plan on running an ablation study like this, but should we need to cut down on compute or make our model train faster, we can consider removing some of the least significant modules from our overall model. Finally, the researchers analyzed the values of their three hyperparameter weighting coefficients: *alpha*, *beta*, and *upsilon* on both datasets to verify their selection of values. We will not run any similar experiments on the model itself, but we will be testing our model on additional datasets (SarcNet and MORE).

Goals

Base: Model successfully labels test split with either sarcastic/not sarcastic. Model is implemented with Keras MultiHeadedAttention (no multiscale deformable attention). Goal accuracy of 60%
Target: Accuracy of 78% on MMSD 2.0
Stretch: Model is implemented with multiscale deformable attention

Ethics

The development of sarcasm detection models like RCLMuFN raises ethical considerations regarding their societal impact and potential biases in the underlying datasets. The primary ethical questions of this field involve the utility of accurate sarcasm detection, dataset biases, and broader implications for online communication.

Societal Impact of Sarcasm Detection

Accurate sarcasm detection can have several potential benefits. As the authors of the original paper note, such models can aid in "identifying and filtering undesirable information on the Internet, thereby reducing malicious defamation and rumor-mongering". There is a growing need for nuanced content interpretation in digital spaces, as evidenced by new trends like appending "/s" to indicate sarcastic intent in text-based communications on reddit and other platforms.

The societal value of sarcasm detection manifests in multiple ways. First, it can contribute to misinformation reduction. Sarcastic statements misinterpreted as literal can propagate false information. Schifanella et al. (2016) demonstrated that sarcasm often involves a reversal of sentiment valence, potentially leading to significant misunderstandings when automated systems fail to recognize this reversal. Second, in contexts where sarcasm may be inappropriate or counterproductive, such as crisis communication, medical information sharing, or formal educational settings, detection tools can assist human moderators in identifying potentially problematic content. Third, Maynard and Greenwood (2014) found that incorporating sarcasm detection can significantly improve sentiment analysis accuracy, leading to a 21% performance increase. This has implications for market efficiency as sentiment analysis increasingly informs capital allocation.

Dataset Biases and Limitations

The RCLMuFN model's training on MMSD and MMSD 2.0 datasets introduces several important biases that warrant examination. These datasets contain exclusively English content, limiting the model's abilities across languages due to different sentence structures and cultural norms. Sarcasm presentation and interpretation vary substantially across cultural contexts: for example British sarcasm tends toward subtlety, while American sarcasm often employs more overt markers. The human labeling process itself introduces potential biases, as annotators will have their own interpretations of sarcasm, which is quite subjective.

Broader Ethical Implications

Beyond the immediate applications and biases, several broader ethical considerations deserve attention. The determination of when sarcasm is "inappropriate" involves subjective judgments that may reflect particular cultural or institutional values. Implementing sarcasm detection for content moderation must consider whose standards of appropriateness are being enforced. As global online communication increases, models trained primarily on Western data may misinterpret sarcastic expressions from non-Western cultures, potentially reinforcing existing power imbalances in global discourse. Any implementation of sarcasm detection should maintain human oversight, particularly in high-stakes contexts where misclassification could have significant consequences for individuals or communities.

Future work should focus on developing more culturally diverse datasets, exploring cross-linguistic sarcasm detection, and establishing ethical guidelines for the responsible deployment of these technologies in various contexts.

References:

Schifanella, R., de Juan, P., Tetreault, J., & Cao, L. (2016). Detecting Sarcasm in Multimodal Social Platforms. In Proceedings of the 2016 ACM on Multimedia Conference.

Maynard, D., & Greenwood, M. A. (2014). Who cares about sarcastic tweets? Investigating the impact of sarcasm on sentiment analysis. In Proceedings of LREC 2014.

Division of labor

We are all planning to work on the model structure together and incorporating the separate features into the model

Bold: stuff to work on next

- Catherine
  - Loading and preprocessing MMSD data
  - train.py and predict.py
  - model.py
- Jake
  - transforms and box_ops
  - **Processing new dataset (SarcNet or MORE)**
  - model.py
- Isabelle
  - How to run code with Oscar
  - Loading and preprocessing MMSD data
  - model.py
- Aaron
  - Ethics + Data in paper
  - **backbone** and position encoding
  - model.py

- **Multiscale deformable attention**