Multi-Modal Machine Learning Systems

for Social Media Content Moderation with Dialogue Summarization and Argument Graphs

Faiaz S. Rahman
faiaz.rahman@yale.edu

A thesis submitted in partial fulfillment
of the requirements for the degree
Bachelor of Science in Computer Science

May 2022
Advised by Dr. Dragomir R. Radev
Department of Computer Science

Yale University
New Haven, Connecticut

This thesis is dedicated to my parents, Dalia and Akm

Abstract

The proliferation of misinformation and hate speech online has created an era of digital disinformation, public mistrust, and even violence, particularly on social media platforms where users can engage in dialogue with such content. Fake news and hate speech exist not only in text form, but also include any accompanying images and video with the original post. Multi-modal models (i.e., those incorporating multiple modalities of data like text and images) offer a powerful approach in detecting such content. Prior work has both developed hate and misinformation datasets for experimentation, and examined different multi-modal representations in general, particularly for text–image data.

Given that user dialogue (e.g., comment threads, Tweet replies, etc.) can often give more insight into the integrity or hatefulness of a post (e.g., by indicating how extreme of a response was garnered, by introducing viewpoints beyond those of the original author, etc.), we investigate methods for modeling and incorporating the dialogue modality into multi-modal models. Specifically, we (1) develop multi-modal models for content moderation tasks (for the modalities of text, image, and dialogue), (2) improve dialogue modeling within those models by introducing ArgSum, an argument graph–based approach to dialogue summarization, and (3) improve the modeling of cross-modal interactions through the multi-modal fusion methods of uni-modal early fusion and low-rank tensor fusion.

Our experiments find that (1) the incorporation of the dialogue modality in multi-modal models improves performance on fake news detection, (2) modeling argumentative structures in dialogues via ArgSum improves both summarization quality and multi-modal model performance, and (3) low-rank tensor fusion is able to better model cross-modal interactions than early fusion. Additionally, we release a public codebase including all of our PyTorch models, our ArgSum software package, and our experiment configuration files, built with an extensible design for future work on hate and misinformation detection.[1]

[1]For reproducibility and extensibility in future work, we make our code, which includes our PyTorch models, an automated process for running experiments, and documentation on how to use the ArgSum software package, available at https://github.com/faiazrahman/Multimodal-Content-Moderation.

Contents

1 Introduction

The proliferation of misinformation and hate speech online has created an era of digital disinformation, public mistrust, and even violence, particularly on social media platforms where users can engage in dialogue with such content. According to the 2021 Pew Research Center survey[2], 86% of adults in the U.S. get their news from a digital device (e.g., a smartphone, computer, or tablet) and 53% of adults in the U.S. get news from social media. As a result of its provocative design, fake news spreads more frequently and more quickly when compared to true, fact-based news (Vosoughi et al., 2018). On these online platforms, hate speech also permeates, reproducing systemic racism and existing prejudice (Matamoros-Fernández and Farkas, 2021). Both fake news and hate speech can be amplified exponentially through user sharing (Talwar et al., 2020; Allcott et al., 2019) and recommendation algorithms, often more quickly than normal content (Mathew et al., 2019), making their detection paramount.

Fake news and hate speech exist not only in text form, but also include any accompanying images and video with the original post. Multi-modal models (i.e., those incorporating multiple modalities of data like text and images) offer a powerful approach in detecting such content. Prior work has both developed hate and misinformation datasets for experimentation, and examined different multi-modal representations in general, particularly for text–image data.

Given that user dialogue (e.g., comment threads, Tweet replies, etc.) can often give more insight into the integrity or hatefulness of a post (e.g., by indicating how extreme of a response was garnered, by introducing viewpoints beyond those of the original author, etc.), we investigate methods for modeling and incorporating the dialogue modality into multi-modal models.

Our **problem description** is as follows: How can multi-modal models incorporate the dialogue modality to improve performance on content moderation tasks? In this process, are there ways to leverage the argumentative structures inherent in dialogues? Once incorporated, are there methods which can improve the modeling of the cross-modal interactions between the different data modalities?

In response to these research questions, we (1) develop multi-modal models for content moderation tasks (for the modalities of text, image, and dialogue), (2) improve dialogue modeling within those models by introducing ArgSum, an argument graph–based approach to dialogue summarization, and (3) improve the modeling of cross-modal interactions through the multi-modal fusion methods of uni-modal early fusion and low-rank tensor fusion. We design models, implement them in PyTorch, and run experiments to evaluate each of these three objectives.

[2] https://www.pewresearch.org/internet/2021/04/07/social-media-use-in-2021/

2 Related Work

2.1 Multi-Modal Machine Learning

2.1.1 Motivation for Multi-Modal Learning

A *modality*, in more general terms, refers to the way in which something in the world happens or is experienced; as humans, our sensory modalities include vision, taste, and touch. When translating this phenomena into computer science, we view modality as distinct forms of data which each have their own nuances in processing, learning, and inference (Baltrusaitis et al., 2017).

Multi-modal machine learning models are those which can incorporate multiple modalities of data in their processing, including text, image, audio, video, and dialogue, and even body gestures, facial expressions, and so on (Summaira et al., 2021). These models are useful for their potential in solving cross-modality tasks, i.e., problems which can better be solved by incorporating the additional information that comes with multiple modalities.

There are five core processes (each bringing its own challenges) in multi-modal machine learning: (i) *representation* of multi-modal data that exploits complementarity and reduces redundancy between multiple modalities, made difficult by the heterogeneity of data (Guo et al., 2019); (ii) *translation* from one modality to another, which is often subjective; (iii) *alignment*, or the identification of direct relations between elements of different modalities; (iv) *fusion* of modalities when performing prediction (Zhang et al., 2020); and (v) *co-learning*, or the transfer of knowledge between modalities, their representation, and their predictive models (Baltrusaitis et al., 2017).

Despite the quixotic intuition for multi-modal models to outperform standard machine learning models given their incorporation of various modalities of data, often times, uni-modal models can actually outperform their multi-modal counterparts, given that (i) multi-modal models can be prone to overfitting due to their increased capacity and (ii) different modalities overfit and generalize at different rates, making joint training difficult (Wang et al., 2019). Some modalities' data can also be sparser or noisier than that of others, hindering model performance. As a result, multi-modal machine learning must be approached carefully.

2.1.2 Cross-Modal Interactions and Multi-Modal Fusion

In multi-modal machine learning tasks, different modalities interact with one another, which we define as *cross-modal interactions*; given that these interactions could either improve or hinder model performance, understanding them can better inform multi-modal model architecture design (Hessel and Lee, 2020). If the cross-modal interactions are actually hindering model performance, it may be the case that the multiple views of data are not in agreement when making predictions (Ding and Tibshirani, 2021).

Knowledge of cross-modal interactions helps better facilitate *multi-modal co-learning*, in which we aim for transferring useful information between modalities for not only improved prediction performance, but also model complexity and interpretability (Pham et al., 2018; Tan and Bansal, 2020). Co-learning is especially important in real-world tasks where one or more modalities is missing, noisy, lacking labeled data; in such scenarios of data modality scarcity, resource-poor modalities benefit from information transfer from resource-rich modalities (Pahde et al., 2020; Ma et al., 2021; Rahate et al., 2022).

Cross-modal interactions can be introduced through various *multi-modal fusion* techniques; we describe and experiment with two fusion methods in Section 6.

2.1.3 Leveraging Pretraining for Multi-Modal Models

Several landmark multi-modal models have been developed in recent years, all of which emphasize pretraining. VisualBERT (Li et al., 2019) was a simple but significant multi-modal model for

vision and language which incorporated BERT-like pretraining; specifically, it employed (i) *masked-language modeling with the image* where some tokens were masked (but the corresponding image vectors were not) and (ii) *sentence-image prediction* in which it was predicted whether a caption corresponded to an image.

These BERT-like pretraining objective inspired future work, including Oscar (Li et al., 2020), which went further to align image objects with text semantics. Specifically, Oscar used Faster R-CNN to detect a set of object tags in the image to use as anchor points; thus, a word–tag–image triple (w, q, v) was passed as input to the model, with masked token loss and contrastive loss as its pretraining objectives. Oscar succeeded on image retrieval, text retrieval, image captioning, novel object captioning, visual question answering, and the "Natural Language Visual Reasoning for Real" task, with improved intra-class and inter-class properties in its learned semantic feature space. Oscar was later further improved by VinVL (Zhang et al., 2021), specifically for better visual representations.

More recently, Facebook AI developed FLAVA (Singh et al., 2021), a foundation language and vision alignment model aimed to tackle uni-modal, cross-modal, and multi-modal tasks. FLAVA incorporated an image encoder, a text encoder, and a multi-modal encoder (all of which were Transformer-based) with several pretraining objectives including masked image modeling (MIM), masked language modeling (MLM), contrastive, masked multi-modal modeling (MMM), and image-text matching (ITM) losses. This pretraining was conducted using publicly-available multi-modal datasets, and the resulting model had impressive performance on a wide range of 35 tasks, illustrating the power of multi-modal models to truly become foundation models.

2.2 Content Moderation Systems

Fake news on the Internet, and particularly social media, is dangerous for creating public mistrust and pushing consumers to accept false beliefs which forward specific agendas (Shu et al., 2017). In a similarly dangerous vein, hate speech online reproduces systemic racism (Matamoros-Fernández and Farkas, 2021), targets marginalized groups, and can contribute to horrifying outcomes, like the Rohingya genocide in Myanmar (Mathew et al., 2019). The current COVID-19 pandemic is even shaped by a fake news "infodemic" (van der Linden et al., 2020) and the weaponization of COVID-19 to fuel hate speech and racism online (Velásquez et al., 2021).

Fake news and hate speech can be amplified exponentially through user sharing (Talwar et al., 2020; Allcott et al., 2019) and recommendation algorithms, often more quickly than normal content (Mathew et al., 2019), making their detection paramount. Shu et al. (2017) argues that fake news detection is difficult when based solely on news content, since it is intentionally written to mislead users, and thus *auxiliary information* such as user social engagements is needed; in our work, user dialogue takes this role.

2.2.1 Fake News Detection Models

Previous work on fake news detection has involved linear SVM classifiers (Pérez-Rosas et al., 2018), models judging fake news based on their reasoning (Hansen et al., 2021), meta-learning to detect fake news on newly-emerging events with few verified posts (Wang et al., 2021), and multi-source (e.g., a statement, metadata, history, and a report) frameworks (Karimi et al., 2018). There are many fake news datasets available (Wang, 2017; Shu et al., 2019; Thorne et al., 2018); however, the majority of the datasets consist of only text-based data. Fakeddit (Nakamura et al., 2020), the dataset we use for the fake news detection task, is a large multi-modal dataset with text, image, and dialogue data with multiple k-way classification labels.

2.2.2 Hate Speech Detection Models

Previous work on hate speech detection includes HateBERT (Caselli et al., 2021), which fine-tuned the BERT language model (Devlin et al., 2019) for abusive language detection, and neural models integrating text semantics with socio-cultural context (Vijayaraghavan et al., 2021). Several multimodal (specifically, text and image) hate speech datasets include Facebook AI's Hateful Memes (Kiela et al., 2021), MMHS150K (Gomez et al., 2019), MultiOFF (Suryawanshi et al., 2020), and ALONE (Wijesiriwardene et al., 2020); it is important to note, however, that these datasets themselves do not have dialogue data, but are derived from social media platforms with user dialogues (e.g., Twitter and its tweet replies); thus, there is the potential to augment these datasets with their dialogue data.

2.3 Conversational Dialogues as a Modality of Data

Dialogue on social media allows for users to engage with both posts and one another, which ideally would result in a communication inclusive of multiple voices which can facilitate understanding rather than persuasion (Jameson and Lee, 2020). Although this ideal scenario may not always hold, dialogue still presents a powerful modality of data from parties other than a post's original author, which in turn can give a measure of the post's reception, integrity, or even hatefulness. We view a dialogue as a set of *utterances* which are linked through responses and subthreads.

2.4 Argument Mining and Argument Graph Construction

Prior work in argument mining and argumentative structure modeling has included the argumentative claim parsing approach of Boltužić and Šnajder (2020) (further subdivided into claim segmentation and claim structuring), the ArgumenText approach's notion of argumentative structures of claims, premises, and argument relations (Daxenberger et al., 2020), and the issues–viewpoints–assertions framework of Barker and Gaizauskas (2016).

Lenz et al. (2020) describe an argument mining pipeline which can be leveraged for constructing argument graphs, i.e., graph structures representing argumentative unit nodes and the relations between them.

2.5 Text and Dialogue Summarization

Automatic text summarization aims to output the most salient parts of a given corpus, either in an extractive manner (where salient portions of the given corpus are extracted and pieced together into a summary) or an abstractive manner (where the model generates text of its own, e.g., as if it were writing "in its own words"). Dialogue summarization, or conversation summarization, aims to summarize a set of utterances. Gliwa et al. (2019) introduced SAMSum, a chat-dialogue dataset for the conversation summarization task. Fabbri et al. (2021) introduced ConvoSumm, a dataset covering four dialogue domains of news article comments, discussion forums, community question-answering, and email threads.

3 Data

We employ five datasets, which can be grouped into three categories: (i) data used for the content moderation tasks: Fakeddit (Nakamura et al., 2020) and MMHS150K (Gomez et al., 2019); (ii) data used to train the submodels in the ArgSum algorithm (prior to its incorporation in the multi-modal models): AMPERSAND (Chakrabarty et al., 2019), Stab & Gurevych's argument mining dataset[3] (Stab and Gurevych, 2014), and MNLI (Williams et al., 2018); and (iii) data used to evaluate ArgSum's general performance via summarization metrics and fine-tune Transformer models for dialogue summarization for usage in the content moderation tasks: SAMSum (Gliwa et al., 2019).

We have a data preprocessing pipeline for each of these three data categories; this is described in more detail in Appendix A.

3.1 Multi-Modal Fake News Detection: Fakeddit

Fakeddit (Nakamura et al., 2020) is a multi-modal dataset consisting of over 1 million samples from multiple categories of fake news, labeled with 2-way (true, fake), 3-way (true, fake with true text, fake with false text), and 6-way (true, satire, false connection, imposter content, manipulated content, misleading content) classification categories to allow for both binary classification and, more interestingly, fine-grained classification. The dataset was collected from a diverse array of topic categories (i.e., subreddits) from Reddit, and includes a post's text, image, and comment threads. We used a randomly-sampled subset of their train and test datasets with a balanced class distribution (and selecting only examples which were multi-modal, i.e., with text, image, and comment data), consisting of 10,000 training examples and 1,000 evaluation examples.

Nakamura et al. (2020) ran experiments comparing multi-modal input of text and image data with single-modal input of text data and image data individually, finding that the multi-modal approach of using text and images simultaneously improved performance. However, despite collecting the comment thread data, the authors did not run any experiments integrating the comment thread data into their input representations, instead leaving the comment data (and additional metadata) for future work, which we pick up from.

The Fakeddit dataset is made publicly available[4] for usage; our codebase has a slightly modified version of their image downloading script.

3.2 Multi-Modal Hate Speech Detection: MMHS150K

MMHS150K (Gomez et al., 2019) is a multi-modal dataset consisting of 150,000 samples of online hate speech (containing text, image, and OCR)[5], labeled with 6-way classification categories (not hate speech, racist, sexist, homophobic, hate towards religion, and other types of hate). The dataset was collected from Twitter, with the tweets having been posted from September 2018 to February 2019. Tweets were collected in real-time (prior to Twitter's own hate speech filters and moderation being applied, which the authors report at the time of their data collection was based on user reports and thus not instantaneous) and then filtered (e.g., to remove retweets, tweets containing less than three words, and tweets without images).

Gomez et al. (2019) ran experiments using a feature concatenation model (using an LSTM for text data and a CNN for image data) and a textual kernals model (to "learn kernels dependent on the textual representations and convolve them with the visual representations in the CNN"), but found that incorporating both modalities did not significantly improve classification accuracy. We posit that this was due to the relative model simplicity of using an LSTM and CNN, and in turn base our text encoders on pretrained Transformer models and our image encoders on ResNet and

[3] In some data preprocessing files in our codebase, this is abbreviated as `SGAM`.

[4] https://github.com/entitize/Fakeddit

[5] OCR refers to the text within an image, which was extracted using Optical Character Recognition.

pretrained Vision Transformer models.

The MMHS150K data is made publicly available[6] for usage; our codebase includes a lightweight `requirements.txt` file for creating a virtual environment, installing the dependencies via the `pip` package manager, and downloading the dataset via the `kaggle` command-line tool.

Additionally, we attempted to augment the original MMHS150K with dialogue data (specifically by scraping tweet replies to the original tweets and building a dialogue thread for each), using the Twitter API, but found that the Twitter API does not allow access to the tweet objects of suspended users (which was the case for all the hateful tweets). As a result, we use Fakeddit data for all experiments involving dialogue data, and use MMHS150K for selected text–image experiments.

3.3 Argumentative Unit Classification: AMPERSAND and Stab & Gurevych

For argumentative unit classification (i.e., the task of classifying an argumentative unit as a claim, premise, or non-argumentative unit; described in more detail in 5.2.1), we use aggregate data from two datasets.

AMPERSAND (Chakrabarty et al., 2019) is a dataset annotated with argumentative components of claims, premises, and major claims, along with intra-turn and inter-turn argumentative relations. We use the argumentative components data, filtering it to keep claims, premises, and non-argumentative units.

Stab and Gurevych (2014) introduced a dataset consisting of argumentative structures in persuasive essays. As before, we filter the data to keep claims and premises (and re-label major claims as claims, since ArgSum determines "major claims" not from classification, but rather during its graph construction process as claims which do not entail any other claims and thus form a subtree directly connected to the graph's root node, described in more detail in 5.2.2).

The AMPERSAND data[7] and the Stab & Gurevych data[8] are both made publicly available. In our codebase, we provide a `README` for both datasets outlining the relevant data files to download and describing the data format. Our data preprocessing pipeline prepares, filters, and aggregates both datasets into a single dataset which can then be used directly by the PyTorch `torch.utils.data.Dataset` for the argumentative unit classification model.

3.4 Textual Entailment for Relationship Type Classification: MNLI

For relationship type classification (i.e., the task of classifying the directed edge from one argumentative unit node to another as supporting, contradicting, or neutral, described in more detail in 5.2.2), we use MNLI data. MNLI (Williams et al., 2018) is a dataset consisting of 433,000 sentence pairs annotated with textual entailment information (specifically, with labels of `ENTAILMENT`, `CONTRADICTION`, and `NEUTRAL`).

The MNLI data is made publicly available.[9] Our codebase includes a Bash script for downloading and processing it.

3.5 Dialogue Summarization: SAMSum

SAMSum (Gliwa et al., 2019) contains over 16,000 chat dialogues and their abstractive summaries. It is made publicly available as a `7zip` on arXiv[10], and also via TensorFlow[11] and Hugging Face[12]. Our codebase includes a Bash script for downloading it from arXiv and processing it.

[6] https://www.kaggle.com/datasets/victorcallejasf/multimodal-hate-speech
[7] https://github.com/tuhinjubcse/AMPERSAND-EMNLP2019
[8] https://github.com/textmining-project/ArgumentMining-Backend
[9] https://cims.nyu.edu/~sbowman/multinli/multinli_1.0.zip
[10] https://arxiv.org/src/1911.12237v2/anc/corpus.7z
[11] https://www.tensorflow.org/datasets/catalog/samsum
[12] https://huggingface.co/datasets/samsum

4 Multi-Modal Models for Content Moderation

We now discuss our methods and approach for developing multi-modal models for content moderation. This section details the specific multi-modal model architectures we employ, while also providing brief background on each of the encoder models and how they are applied in our architecture.

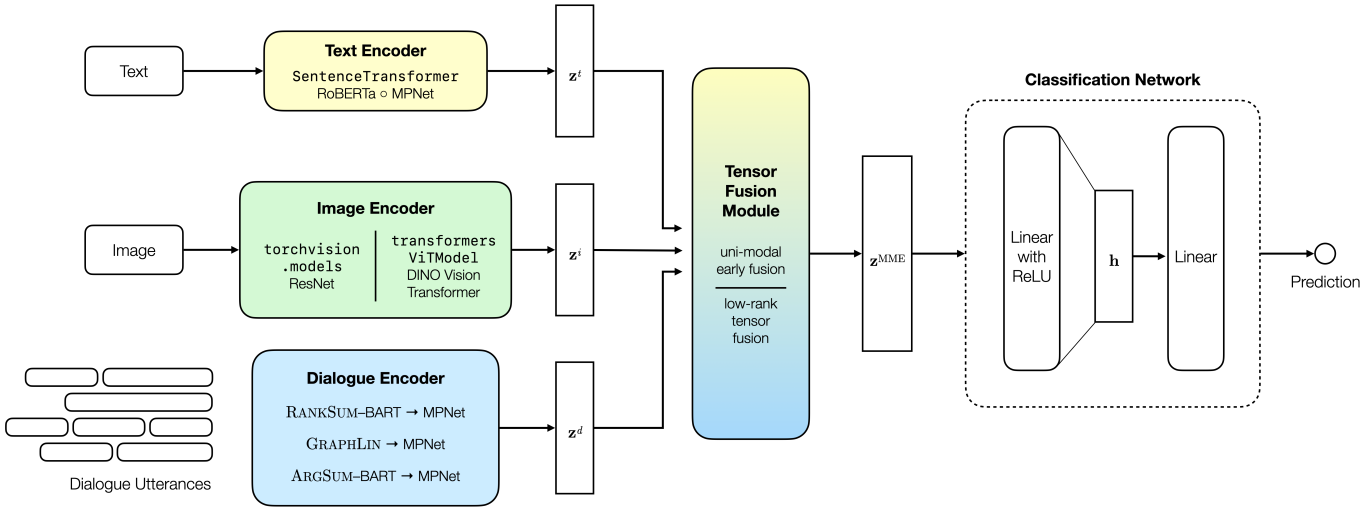

Figure 1: The general architecture for our multi-modal models, consisting of encoders for each data modality, a tensor fusion module, and a classification network. Note that we experiment with multiple models for each component.

4.1 Overview of Model Architecture

Our multi-modal models consist of the following:

(i) *encoders* for each data modality (i.e., text, image, and dialogue), which generate an embedding tensor for each modality;

(ii) a *tensor fusion module*, which fuses the uni-modal representations into a joint multi-modal embedding; and

(iii) a *classification network*, which the multi-modal embedding is then passed through for the content moderation tasks.

In our implementation, all model parameters are kept as trainable, and thus the entire multi-modal model architecture can be trained (thus fine-tuning representations for the specific content moderation task at hand).

4.2 Text Embeddings

We use Transformer-based models to generate text embeddings. We first describe Transformers and the specific models we use, and then our text embedding implementation in the following.

4.2.1 Models

Transformers The Transformer (Vaswani et al., 2017) is a deep learning network architecture based on self-attention mechanisms, designed to process sequential data and replacing RNN-based models such as LSTMs and GRUs. Transformers do not process the data in-order, given that the self-attention mechanism gives context to words in the sequence, thus increasing training parallelizability and leading to the development of pretrained language models such as BERT (Devlin et al., 2019), BART (Lewis et al., 2020), T5 (Raffel et al., 2020), and GPT (Brown et al., 2020).

RoBERTa RoBERTa (Liu et al., 2019) retrains the original BERT bidirectional Transformer model with a robustly optimized training process and more compute. This includes changing BERT's next sentence prediction (NSP) pretraining objective to dynamic masking (but it maintains BERT's masked language modeling, or MLM).

MPNet MPNet (Song et al., 2020) aims to solve the issue of BERT neglecting dependency among predicted tokens through permuted language modeling (thus replacing BERT's masked language modeling) and also takes auxiliary position information as input.

4.2.2 Implementation

The raw input text is passed through a Transformer-based encoder to generate an embedding; our implementation takes variable-length input text and produces a fixed-size text embedding. We experiment with both RoBERTa and MPNet as our text encoder models.

Let v_t^* denote variable-length raw text size and d_t the text embedding dimension.

$$\texttt{EncodeText:} \quad \mathbb{R}^{v_t^*} \mapsto \mathbb{R}^{d_t}$$

4.3 Image Embeddings

We experiment with two approaches for generating image embeddings: ResNet and DINO-based Vision Transformers.

4.3.1 ResNet

In our first approach, the raw image input is first preprocessed into RGB channels, then resized and normalized to fixed dimensions. The preprocessed image tensor is then passed to ResNet (He et al., 2015); specifically, we use ResNet-152. The last layer of ResNet is overwritten with a fully-connected linear layer to get the image features (rather than a classification).

Let v_i^* denote variable-length raw image size, d_n represent the resized and normalized image dimension for one channel (i.e. RGB), and d_i the image embedding dimension.

$$\texttt{EncodeImage (ResNet):} \quad \mathbb{R}^{v_i^*} \mapsto \mathbb{R}^{d_n \times d_n \times d_n} \mapsto \mathbb{R}^{d_i}$$

4.3.2 DINO-based Vision Transformers

In our second approach, we use DINO-based Vision Transformers. The motivation for using a Vision Transformer (ViT) is that it has been shown to outperform ResNet and simpler CNN-based models for pure vision tasks (Dosovitskiy et al., 2020). The motivation for using a DINO-based ViT is its ability to learn features which contain explicit information about images' semantic segmentation (Caron et al., 2021). We describe the DINO algorithm, the Vision Transformer architecture, and our implementation in the following.

DINO: Self-supervised learning with knowledge distillation Caron et al. (2021) developed DINO as a self-supervised method incorporating knowledge distillation for training Vision Transformers. *Knowledge distillation* is a learning paradigm where a student network g_{θ_s} parameterized by θ_s is trained to match the output of a given teacher network g_{θ_t} parameterized by θ_t. Given an input image x, both networks output probability distributions P_s and P_t over K dimensions. The output of the network g is normalized with a softmax function to produce the probability P, as follows (where τ_s and τ_t are temperature parameters that control the sharpness of the output distributions).

$$P_s(x)^{(i)} = \frac{\exp(g_{\theta_s}, (x)^{(i)}/\tau_s}{\sum_{k=1}^{K} \exp(g_{\theta_s}(x)^{(k)}/\tau_s)} \tag{1}$$

$$P_t(x)^{(i)} = \frac{\exp(g_{\theta_t}, (x)^{(i)}/\tau_t}{\sum_{k=1}^{K} \exp(g_{\theta_t}(x)^{(k)}/\tau_t)} \tag{2}$$

Thus, the student network g_{θ_s} learns to match its distributions to that of g_{θ_t} by minimizing the cross-entropy loss.

$$\min_{\theta_s} H(P_t(x), P_s(x)) \tag{3}$$

Caron et al. (2021) adapt this knowledge distillation problem to *self-supervised learning* by constructing different distorted views (also referred to as crops) of an image by applying a set of transforms to the original image, producing both *global views* and *local views* of smaller resolution. The loss is thus adapted as follows.

$$\min_{\theta_s} \sum_{x \in \{x_1^g, x_2^g\}} \sum_{x' \in V, x' \neq x} H(P_t(x), P_s(x')) \tag{4}$$

Thus, the model can be pretrained on unlabeled image data by generating these distorted transformations of the original images.

Both networks share the same architecture g with separate sets of parameters θ_s, θ_t. Since this is self-supervised learning, there is no a priori for the teacher network, and thus it is built from past iterations of the student network (e.g., freezing the teacher network over an epoch or a specified number of training steps).

Vision Transformers (ViTs) The architecture g consists of a backbone f (in our experiments, we use ViT) and a projection head $h : g = h \circ f$. ViT (Dosovitskiy et al., 2020) takes as input a grid of non-overlapping contiguous $N \times N$ image patches, passes them through a linear layer to generate embeddings, adds a `[CLS]` token, and feeds them into a standard Transformer network, i.e., a sequence of self-attention and feedforward layers parallelized with skip connections (Vaswani et al., 2017).

Implementation In our DINO ViT image encoder, the raw image input is again first preprocessed into RGB channels and then resized to match the expected input dimensions for the Vision Transformer model. As described previously, the image is divided into $N \times N$ patches, passed through a linear layer to generate embeddings, added with a `[CLS]` token, and fed into a standard Transformer network. The final hidden states of the DINO ViT are used as the image embedding, i.e. $\in \mathbb{R}^{d_i}$.

Let v_i^* denote variable-length raw image size, d_n represent the resized and normalized image dimension for one channel (i.e. RGB), d_i the image embedding dimension, and N the patch size for the ViT. Also, let P_i be a single patch.

$$\texttt{EncodeImage (DINO ViT):}\ \mathbb{R}^{v_i^*} \mapsto \mathbb{R}^{d_n \times d_n \times d_n} \mapsto [P_0, \ldots, P_n : P_i \in \mathbb{R}^{N \times N}] \mapsto \mathbb{R}^{d_i}$$

4.4 Dialogue Embeddings

Dialogue data for a single example consists of a set of utterances. In this section, we outline our initial method for embedding dialogues, through a ranked abstractive text summarization process. (In Section 5, we describe the ArgSum algorithm for improving dialogue summarization through the use of argument graphs; the way in which ArgSum is incorporated to this multi-modal model architecture is described in 5.3.3.)

4.4.1 RankSum: Ranked Dialogue Summarization

For our fake news detection data, the raw dialogue consists of a set of comments associated with each post. For our initial dialogue summarization approach, which we title as RankSum (to differentiate it from the approaches described in Section 5), we rank the comments by the number of up-votes they receive (which is included in the metadata), sorting them in descending order and passing the sorted sequence into a Transformers-based summarization pipeline; for our experiments, we use a BART summarization pipeline. We also note that some posts have a large number of comments, which in turn exceeded the input token limit of the summarization pipeline. In that case, we simply truncate comments beyond that limit, and given our sorting method, those are comments which likely have less salience to the overall summary (as measured by the upvotes metadata).

The generated output summary text is then passed through a Transformer-based encoder to get the final dialogue summary embedding; we again experiment with both RoBERTa and MPNet as our text encoder models.

Let v_d^* denote variable-length raw dialogue utterance size, d_d the dialogue embedding dimension.

$$\texttt{EncodeDialogue (RankSum):} \quad [\mathbb{R}^{v_d^*}, ...] \mapsto \mathbb{R}^{d_d}$$

We also note that when generating our summaries, we determine our generated summary's minimum and maximum length bounds by the following heuristic (computed sequentially as shown).

$$n = \sum_{w \in \text{utterance}} 1$$
$$\text{max\_length} = \min\left(75, \frac{n}{2}\right)$$
$$\text{max\_length} = \max(\text{max\_length}, 5)$$
$$\text{min\_length} = \min(5, \text{max\_length} - 1)$$

This was found empirically to yield good summary quality while avoiding implementation-specific issues of the Transformers pipeline (e.g., a maximum length of 1 yields unexpected behavior, hence the second update of the maximum length variable).

4.5 Tensor Fusion Module for Multi-Modal Embedding

Once the individual embeddings are obtained, they are combined in a tensor fusion module. For uni-modal early fusion, we apply tensor concatenation to the individual embeddings and pass it to a feedforward fully-connected layer (followed by ReLU); the output of the feedforward layer is the final multimodal embedding, which is then used in the classification pipeline.

Let d_{MME} denote the final multi-modal embedding dimension.

$$\texttt{MultimodalEmbed (Early Fusion):} \quad \mathbb{R}^{d_t} \times \mathbb{R}^{d_i} \times \mathbb{R}^{d_d} \mapsto \mathbb{R}^{d_t+d_i+d_d} \mapsto \mathbb{R}^{d_{\text{MME}}}$$

Additional fusion methods we experimented with are described in Section 6.

4.6 Classification Network for Content Moderation Tasks

The multi-modal embedding is then passed through two fully-connected feedforward layers, i.e., mapping from the embedding to a hidden layer, and then to the final output layer with k nodes, where k is the k-way classification being done. (In our experiments, $k \in [6, 3, 2]$.)

Let d_h denote the hidden dimension.

$$\texttt{Classify:} \quad \mathbb{R}^{d_{\text{MME}}} \mapsto \mathbb{R}^{d_h} \mapsto \mathbb{R}^k$$

Thus, the overall multi-modal model architecture allows for modular embedding of the individual modalities prior to their fusion; the resulting multi-modal embedding is then used in the classification portion of the overall model to yield the fine-grained prediction.

5 ArgSum: Argument Graph–Based Dialogue Summarization

The RankSum method described in 4.4 provides an efficient, scalable process for generating dialogue embeddings for our multi-modal content moderation models. However, it focuses on using comment metadata (e.g., upvotes) to rank comments, and does not leverage the semantic structure of the dialogue itself. Furthermore, it requires having the comment metadata in order to produce good ranking. This motivates the exploration of alternative methods for producing representations of dialogue data.

In this section, we present ArgSum, an algorithm for dialogue summarization based on argument graph representations, which does not require any inherent metadata to rank salience and instead focuses on modeling the argumentative structures within the dialogue itself. We give an overview of the algorithm, explain its processes, and then describe how it is incorporated into our multi-modal models. (The pseudocode for ArgSum is presented in Appendix B.)

5.1 Overview of the ArgSum Algorithm

The ArgSum algorithm consists of (1) *argument graph construction*, in which a set of dialogue utterances are constructed into an argument graph (consisting of argumentative unit nodes and relationship type edges), and (2) *argument graph summarization*, in which the resulting graph is linearized and then summarized by a sequence-to-sequence Transformer model.

In our codebase, ArgSum is presented as a standalone software package which has modules to train the submodels for argumentative unit classification and relationship type classification (developed in PyTorch) and data structures to represent argument graphs (in Python 3). This, in turn, can be used to construct argument graphs, linearize argument graphs, and run summarization.

ArgSum is modeled after ConvoSumm's argument graph summarization process (Fabbri et al., 2021), and is motivated by prior work in argument mining and argumentative structure modeling, including the argumentative claim parsing approach of Boltužić and Šnajder (2020) (further subdivided into claim segmentation and claim structuring), the ArgumenText approach's notion of argumentative structures of claims, premises, and argument relations (Daxenberger et al., 2020), and the issues–viewpoints–assertions framework of Barker and Gaizauskas (2016).

5.2 Argument Graph Construction

Extending the argument mining pipeline described in Lenz et al. (2020), we define the process for constructing an argument graph as follows.

We formulate our argument graphs with the following notation. Let A be the alphabet of valid characters (e.g., Unicode characters). Let $D = \{s \in A^n : n \in [1, \infty)\}$ represent the set of dialogue utterances, where each utterance is a string of characters of minimum length 1.

5.2.1 Argument Extraction

Utterance segmentation First, we segment each utterance into its *argumentative units*. Let $U = \{s[i, j] : s \in D, 0 \leq i < \text{len}(s), 0 \leq j < \text{len}(s)\}$ represent the set of segmented utterances, where each segmented utterance (i.e., argumentative unit) is a substring of some original utterance. In our implementation, we use sentence-level segmentation, similar to Lenz et al. (2020).

Argumentative unit classification For each argumentative unit $u \in U$, we classify it as a *claim* (an assertion that something is true), a *premise* (a proposition from which a claim can be made), or a *non-argumentative unit*. We train a model for this classification task using data from AMPERSAND (Chakrabarty et al., 2019) and Stab and Gurevych (2014). (In our codebase, this is denoted as the

AUC model; argumentative units are passed in batches through the trained AUC model to reduce overall inference time.)

This step creates the set V of argumentative unit nodes.

5.2.2 Relationship Type Classification

Once the argumentative units are classified, we then use entailment to determine the relationship between argumentative units, in which an argumentative unit can either support (i.e., entail), contradict, or be neutral to another argumentative unit. We train a model for this task using data from MNLI (Williams et al., 2018). (In our codebase, this is the RTC model.)

Premise-to-claim entailment We first run the RTC entailment model for each premise to all possible claims, selecting the claim node with which it has the highest entailment score for the relationship type `SUPPORTS`. (We define the *entailment score* simply as the classification probability for the predicted label, where the probabilities are computed as the softmax over the logits produced by the model's classification head.) If that entailment score exceeds the minimum threshold, we create an edge from the premise to the claim.

This step clusters premises with the claims that they entail most directly, creating subtrees of depth 1. (Similarly to our AUC model, we pass argumentative unit pairs in batches to our RTC model to reduce overall inference time.)

Claim-to-claim entailment Next, we run the RTC entailment model among all pairs of claims, in both directions (i.e. for all claims c_i and c_j, we compute the entailment score for c_i `SUPPORTS` c_j and for c_j `SUPPORTS` c_i). After collecting entailment scores for all possible claim pairs, we greedily add support edges according to the greatest entailment scores while no cycles are created in the argument graph. (Note that it was not possible to create cycles in the premise-to-claim entailment step, since there was no possibility of a claim entailing a premise.)

This step connects subtrees' claim nodes to other subtrees, creating hierarchical structures with depth greater than 1.

Root node linking Finally, for all claim nodes which do not entail any other claims, we connect them directly to the root node of the graph. (The root node's classification type is neither `CLAIM` nor `PREMISE`, but rather `ROOT_NODE`.) Additionally, for premises that did not entail any claims, we also connect them directly to the root node. The edges connecting these nodes directly to the root are not of relationship type `SUPPORTS`, but rather `TO_ROOT`.

This step connects the entire argument graph, creating a single connected component. The set of edges E thus includes edges produced by both entailment steps and the root node linking step. We formally define the resulting argument graph as $G = (V, E)$.

5.3 Argument Graph Summarization

Once the argument graph is constructed, we have the ability to then linearize it. This linearized argument graph can then be left as is (which is the output of the GraphLin step), or subsequently passed to a sequence-to-sequence Transformer model for text summarization (which is the final output of the ArgSum algorithm).

5.3.1 GraphLin: Argument Graph Linearization

Definitions and invariants of the GraphLin algorithm We define a *claim subtree* as the subtree formed by having a given claim node as root (i.e., with its child claims and child premises). Note that a claim subtree can have other claim subtrees within it. We define the act of *placing* a

node in the linearized sequence as simply adding its segmented utterance text into the linearized text string. From the argument graph construction process described in 5.2, we also note the following invariants: (i) premises are always leaf nodes; (ii) claims can be either internal or leaf nodes; (iii) the root of the tree is a root node (i.e., neither a claim nor a premise).

Heuristics We use the following heuristics for linearizing the argument graph. These heuristics are applied depth-first from the root node.

- **Greedy Claim Placement:** During the graph traversal, the current claim node is immediately added to the linearized text string.
- **Semantic Ordering:** For a given claim node, its child premises are placed before child claims. (This keeps all claims and their immediate premises together, while still ordering subsequent claim subtrees nearby.)
- **Subtree Size Prioritization:** For a given claim node (or the root node), its child claim subtrees are ordered based on decreasing size (i.e., largest subtrees are placed first). (This prioritizes claim subtrees with the most amount of argumentative units — and thus the most salience in the original dialogue.) Note that the claim subtree sizes are precomputed via a vanilla depth-first traversal prior to running the main traversal.
- **Zero-Degree Premise Tailing:** Premises which do not entail any claims (and thus are just connected to the root) are placed at the very end of the entire linearized sequence. (This keeps potentially irrelevant premises away from the front of the sequence, where they will have less weight on the generated summary.)

5.3.2 Linearized Text Summarization

After the argument graph is linearized using GraphLin, we are able to use it directly as input into a sequence-to-sequence Transformer model. In our implementation, we use BART (Lewis et al., 2020), specifically in a text summarization pipeline via Hugging Face `transformers`.

5.3.3 Incorporation of ArgSum into Multi-Modal Models

The modularity of ArgSum's design and implementation allows it to be incorporated into multi-modal models in several ways. The final generated summary can be incorporated as the dialogue modality of a multi-modal model by generating its text embedding; this resulting dialogue embedding can be fused with the embeddings of the other modalities to produce the joint embedding. Similarly, the linearized graph produced by the GraphLin step can also be embedded and fused with the other modalities. Additionally, the argument graph itself could be incorporated using a graph encoder. We run experiments to incorporate both the GraphLin output and the final ArgSum output into our multi-modal models.

6 Multi-Modal Fusion Methods

Section 4 detailed our approach to developing a model architecture for multi-modal content moderation, which included a *tensor fusion module* bridging the uni-modal encoders and the downstream task's classification network; we now detail our methods for this fusion process. For the fusion of data modalities prior to performing predictions, we experiment with two different methods, motivated by the previous work of Zadeh et al. (2017), Liu et al. (2018), and Jayakumar et al. (2020): uni-modal early fusion and low-rank tensor fusion.

6.1 Uni-Modal Early Fusion

For uni-modal early fusion, we apply tensor concatenation to the individual modalities' embedding tensors immediately after they are produced by the individual encoders (Zadeh et al., 2017); then, we pass the resulting tensor to a feedforward fully-connected layer followed by ReLU to generate an embedding in a lower-dimensional space.

This can be applied to any arbitrary number k of data modalities; for our experiments, we use it for two and three modalities, as follows. Let $\mathbf{z}$ represent an embedding tensor. Whereas we used notation specific to text, image, and dialogue modalities in 4.5, let us now more generally let d_a, d_b, and d_c denote the individual embedding dimensions of three different modalities, and let d_{MME} denote the final multi-modal embedding dimension. Let W and b represent the weights and biases of our feedforward fully-connected layer.

$$\mathbf{z}^a \in \mathbb{R}^{d_a}, \mathbf{z}^b \in \mathbb{R}^{d_b} : \mathbf{z}^{\text{MME}} = \text{ReLU}\left(W\left(\left[\mathbf{z}^a, \mathbf{z}^b\right]\right) + b\right)$$
$$\mathbf{z}^a \in \mathbb{R}^{d_a}, \mathbf{z}^b \in \mathbb{R}^{d_b}, \mathbf{z}^c \in \mathbb{R}^{d_c} : \mathbf{z}^{\text{MME}} = \text{ReLU}\left(W\left(\left[\mathbf{z}^a, \mathbf{z}^b, \mathbf{z}^c\right]\right) + b\right)$$

Thus, for our experiments for both fake news detection (with text, image, and dialogue modalities) and hate speech detection (with text, image, and OCR[13] modalities), we have the following.

$$\texttt{MultimodalEmbed (Early Fusion, } k = 2\texttt{):} \quad \mathbb{R}^{d_a} \times \mathbb{R}^{d_b} \mapsto \mathbb{R}^{d_a + d_b} \mapsto \mathbb{R}^{d_{\text{MME}}}$$
$$\texttt{MultimodalEmbed (Early Fusion, } k = 3\texttt{):} \quad \mathbb{R}^{d_a} \times \mathbb{R}^{d_b} \times \mathbb{R}^{d_c} \mapsto \mathbb{R}^{d_a + d_b + d_c} \mapsto \mathbb{R}^{d_{\text{MME}}}$$

Zadeh et al. (2017) define this as "early" fusion since the uni-modal embeddings are concatenated immediately after being produced, distinguishing it from low-rank tensor fusion, which follows.

6.2 Low-Rank Tensor Fusion

6.2.1 Cross-Modal Interactions with Differentiable Outer Products and Reducing Dimensionality

Given that the previous approach concatenates the individual embedding tensors early-on, Zadeh et al. (2017) posit that such methods reduce the ability to model "unimodal, bimodal, and trimodal dynamics" (which is analogous to the cross-modal interactions discussed in 2.1.2). They proposed using a three-fold Cartesian product in which three uni-modal tensors' interactions in pairs and as a triple can be modeled. (Similarly, Jayakumar et al. (2020) propose using layers with *multiplicative interactions* when combining multiple modalities of data.)

For the case of fake news detection, consider a text embedding tensor $\mathbf{z}^t$, an image embedding tensor $\mathbf{z}^i$, and a dialogue embedding tensor $\mathbf{z}^d$. (For hate speech detection, we use the OCR embedding tensor $\mathbf{z}^o$ instead of $\mathbf{z}^d$.) Thus, following Zadeh et al. (2017), we aim to model the following cross-modal interactions:

(i) text–image cross-modal interactions, i.e. $\mathbf{z}^t \otimes \mathbf{z}^i$;

[13] OCR refers to the text within an image, which was extracted using Optical Character Recognition; this only occurs in the hate speech detection dataset we use (see 3.2).

(ii) text–dialogue cross-modal interactions, i.e. $\mathbf{z}^t \otimes \mathbf{z}^d$;

(iii) image–dialogue cross-modal interactions, i.e. $\mathbf{z}^i \otimes \mathbf{z}^d$; and

(iv) text–image–dialogue cross-modal interactions, i.e. $\mathbf{z}^t \otimes \mathbf{z}^i \otimes \mathbf{z}^d$.

This is achieved by computing the *differentiable outer product* between the individual uni-modal embedding tensors. (Note that the additional constant dimension produces all cross-modal interactions; in our implementation, it also constitutes the batch size during training and evaluation.)

$$\mathbf{z}^{\text{MME}} = \begin{bmatrix} \mathbf{z}^t \\ 1 \end{bmatrix} \otimes \begin{bmatrix} \mathbf{z}^i \\ 1 \end{bmatrix} \otimes \begin{bmatrix} \mathbf{z}^d \\ 1 \end{bmatrix}$$

However, as Liu et al. (2018) find, using this tensor directly can be computationally expensive, since the number of parameters needed to then embed $\mathbf{z}^{\text{MME}}$ increases exponentially with each added modality (e.g., using a linear layer $W \cdot \mathbf{z}^{\text{MME}} + b$, W would have a huge number of parameters). Additionally, since this embedding is being trained (i.e., the outer product is differentiable), training the parameters would be similarly expensive.

Thus, Liu et al. (2018) propose methods for *low-rank decomposition* in order to reduce the dimensionality of the uni-modal tensors *prior* to computing their outer product. We adapt this approach with separate linear layers for each modality which reduce the tensors to a smaller dimension, and then compute their differentiable outer product.

6.2.2 Implementation

We found that for bi-modal fusion, dimensionality reduction was not needed since the resulting tensor from the outer product had 90K parameters (which was computationally-feasible). For tri-modal fusion, however, the resulting tensor had 27M parameters without reduction; thus, we applied the aforementioned linear layer dimensionality reduction approach prior to computing the outer product.

Let W_a, W_b, and W_c represent the weight matrix for the dimensionality reduction layers for each modality. Let W (with no subscript) represent the weight matrix for the linear layer taking the tensor produced by the outer product and producing the multi-modal embedding, as before.

$$\mathbf{z}^a \in \mathbb{R}^{d_a}, \mathbf{z}^b \in \mathbb{R}^{d_b} : \mathbf{z}^{\text{MME}} = \text{ReLU}\left(W\left(\mathbf{z}^a \oplus \mathbf{z}^b\right) + b\right)$$
$$\mathbf{z}^a \in \mathbb{R}^{d_a}, \mathbf{z}^b \in \mathbb{R}^{d_b}, \mathbf{z}^c \in \mathbb{R}^{d_c} : \mathbf{z}^{\text{MME}} = \text{ReLU}\left(W\left(\text{ReLU}(W_a(\mathbf{z}^a)) \oplus \text{ReLU}(W_b(\mathbf{z}^b)) \oplus \text{ReLU}(W_c(\mathbf{z}^c))\right) + b\right)$$

Thus, for our experiments for both fake news detection and hate speech detection, we have the following. Let $d_{a'}$, $d_{b'}$, and $d_{b'}$ denote the embedding dimensions after reduction.

$$\texttt{MultimodalEmbed (Low-Rank, } k = \texttt{2):} \quad \mathbb{R}^{d_a} \times \mathbb{R}^{d_b} \mapsto \mathbb{R}^{d_a \times d_b} \mapsto \mathbb{R}^{d_{\text{MME}}}$$
$$\texttt{MultimodalEmbed (Low-Rank, } k = \texttt{3):} \quad \mathbb{R}^{d_a} \times \mathbb{R}^{d_b} \times \mathbb{R}^{d_c} \mapsto \mathbb{R}^{d_{a'}} \times \mathbb{R}^{d_{b'}} \times \mathbb{R}^{d_{c'}} \mapsto \mathbb{R}^{d_{a'} \times d_{b'} \times d_{c'}} \mapsto \mathbb{R}^{d_{\text{MME}}}$$

We implemented the differentiable outer product for both bi-modal and tri-modal settings using PyTorch's `torch.einsum()` Einstein summation notation processor. Since our models process data in batches, we maintained the batch dimension while computing the outer product over the remaining dimensions, i.e., computing the outer product for each example in the batch. For the layer generating the embedding from the outer product tensor, the bi-modal models had 90K parameters and the tri-modal models had 125K parameters, which were both computationally efficient (i.e., did not significantly increase training time) when training with our compute resources and experiment configurations.

7 Experiments

7.1 Overview of Experiments

In implementing the methods discussed in Sections 4, 5, and 6, we conduct experiments to

(1) determine if the incorporation of the dialogue modality is able to improve performance on content moderation tasks (or if it increases noise), comparing multi-modal models against uni-modal and bi-modal baselines;

(2) determine if we can improve the modeling of dialogues through the incorporation of argumentative structures via ArgSum (and to evaluate the dialogue summarization capability of ArgSum in general);

(3) determine if the incorporation of ArgSum's argument graph–based dialogue summaries and GraphLin's linearized argument graphs into multi-modal models can further improve performance on content moderation tasks; and

(4) determine if leveraging cross-modal interactions with the differentiable outer product computations of low-rank tensor fusion improves performance on content moderation tasks (in comparison to the early fusion method).

7.2 Models and Methodology Implementations

7.2.1 Multi-Modal Models and Baselines

For our text encoder, we experiment with both RoBERTa and MPNet. Specifically, our implementation uses `all-distilroberta-v1` and `all-mpnet-base-v2` from `sentence-transformers`. (These same encoders are also used for encoding the images' OCR text in the hate speech detection task.)

For our image modules, we experiment with both ResNet and DINO ViT. Specifically, our implementation uses ResNet-152 via `torchvision.models.resnet152` and `facebook/dino-vitb16` (the base model with patch size 16) via Hugging Face `transformers`' `ViTModel` and `ViTFeatureExtractor`.

For RankSum (after the dialogues are ranked) and ArgSum (after the argument graph is constructed and linearized), we use a BART summarization pipeline from Hugging Face `transformers`, specifically with the `sshleifer/distilbart-cnn-12-6` model. For encoding the generated dialogue summary, we use the same models as the text encoder (i.e., RoBERTa and MPNet).

7.2.2 Training ArgSum's Submodels

The ArgSum algorithm involves two submodels for its graph construction step: one for argumentative unit classification and another for relationship type classification. We experiment with both BERT and RoBERTa with a sequence classification head, using `AutoTokenizer` and `AutoModelForSequenceClassification` from Hugging Face `transformers`, specifically with models `bert-base-uncased` and `roberta-base`.

7.2.3 Incorporating ArgSum and GraphLin into Multi-Modal Models

We experiment with using both the argument graph–based dialogue summary (generated by the ArgSum algorithm) and the linearized argument graph (without summarizing, i.e., stopping after the GraphLin step of ArgSum) to see the effects of modeling argumentative structures on the downstream content moderation tasks.

In our codebase, ArgSum is implemented as a Python software package with an `ArgSum` class. To use it, one can instantiate an instance of the `ArgSum` class with the specific trained submodel versions (for argumentative unit classification and relationship type classification, which is how it loads the trained submodel assets) and other parameters (e.g., the entailment score minimum

threshold); e.g., `argsum = ArgSum(...)`. Then, for GraphLin (i.e., generating a linearized argument graph), one can use the `construct_and_linearize()` method; e.g., `linearized_graph = argsum.construct_and_linearize(dialogue_utterances)`. For ArgSum (i.e., generating a linearized argument graph and summarizing it with a similar BART summarization pipeline as 7.2.1), one can use the `summarize()` method; e.g., `summary = argsum.summarize(dialogue_utterances)`.

This is done during data preprocessing so that during multi-modal model training, the PyTorch dataloaders can get the linearized graph (for GraphLin) or argument graph summary (for ArgSum) for the model to pass directly to a text encoder to generate the final dialogue embedding.

7.2.4 Fusion Experiments

As mentioned in 6.2.2, we use PyTorch's `torch.einsum()` Einstein summation notation processor to compute differentiable outer products in multi-modal models using low-rank fusion.

7.3 Experiment Settings

We run our experiments on 1–4 NVIDIA GeForce RTX 3090 GPUs with Driver version 470.103.01 and CUDA version 11.4. We run some additional experiments on two NVIDIA K80 GPUs with driver version 465.19.01 and CUDA version 11.3. We use Python 3.7, PyTorch (version 1.11.0 with CuDNN version 8.2.0), and PyTorch Lightning (version 1.6.0) to implement our models, including using Lightning trainers for our model training and evaluation. We run our training on multiple GPUs in data parallel (i.e., splitting each batch across all the GPUs specified for the experiment, with each GPU processing an even number of items per batch and the root node aggregating the results). Depending on the experiment, we set the batch size after some tuning to maximize GPU utilization without exceeding CUDA memory.

For most multi-modal models, we used a batch size of 64 (when training on 2 GPUs in parallel, i.e., 32 per GPU), a learning rate of `1e-4`, and train for 2–5 epochs. However, these specific settings changed for some experiments after hyperparameter tuning for better performance; refer to the config files in our codebase for the exact configuration of settings for each experiment. For our multi-modal models and the ArgSum submodels, we use Adam as our optimizer and cross-entropy as our loss function. (We ran hyperparameter tuning with SGD with momentum, but ultimately used Adam for all models.) We save model checkpoints every 100 train steps and at the end of every epoch using PyTorch Lightning callbacks.

7.4 Hyperparameter Tuning for ArgSum Submodels

Our most extensive hyperparameter tuning was for the ArgSum submodels, since they significantly affected graph construction and, in turn, summary quality; we ran trials with base models of `bert-base-uncased` and `roberta-base`, optimizers of Adam and SGD (with momentum 1.0, 0.9, 0.8), and learning rates of `1e-3`, `5e-4`, `3e-4`, `1e-4`, `5e-5`, `3e-5`, and `1e-5`.

We found that for argumentative unit classification (AUC), the two configurations of (`bert-base-uncased`, Adam, `lr=5e-5`) and (`roberta-base`, Adam, `lr=3e-5`) performed the best. For relationship type classification, the configuration of (`bert-base-uncased`, Adam, `lr=3e-5`) performed the best. We also note that RoBERTa did not perform nearly as well as BERT for relationship type classification; we posit that this is because whereas the BERT tokenizer produces `input_ids`, `token_type_ids` (i.e., `0` for the first sequence and `1` for the second sequence), and `attention_mask` tensors, the RoBERTa tokenizer does not use `token_type_ids`. Given that relationship type classification is a sentence-*pair* classification task, having the `token_type_ids` to explicitly indicate the two sentences may have given BERT leverage on the entailment task in particular.

8 Evaluation, Results, and Analysis

We evaluate our methods through two key approaches: (1) we use classification metrics to measure model performance on the content moderation tasks (using the Fakeddit and MMHS150K datasets), following the evaluation approach of prior work on multi-modal content moderation (Nakamura et al., 2020; Gomez et al., 2019), and (2) we use ROUGE (Lin, 2004) to evaluate the summarization quality of ArgSum in general (using the SAMSum dataset). For our classification metrics, our PyTorch Lightning models use `torchmetrics` to compute the accuracy, precision, and recall of each experiment; following the authors of the Fakeddit dataset, we report accuracy as a consistent metric across both 2-class classification and multi-class (i.e., $k = 3, 6$) classification.

8.1 Multi-Modal Models: Evaluation on Detection Tasks

Our first suite of experiments aimed to determine if the addition of the dialogue modality in multi-modal modals which previously only used text and images improves performance on the content moderation tasks, or simply introduces spam and increases noise. The results of our experiments with the text and image uni-modal baselines, the text–image models, and the text–image–dialogue models with RankSum are presented in Table 1.

Modality	**Models**	**Results**		
		2-way	**3-way**	**6-way**
Text	RoBERTa	0.7309	0.7406	0.6127
	MPNet	0.7853	0.7434	0.6189
Image	DINO ViT	0.7043	0.6863	0.6004
	ResNet	0.7260	0.7074	0.6086
Text + Image	RoBERTa + DINO ViT	0.7213	0.6972	0.6871
	MPNet + DINO ViT	0.7971	0.7559	0.6992
	RoBERTa + ResNet	0.8087	0.7816	0.7071
	MPNet + ResNet	0.8232	0.7844	0.7288
Text + Image + Dialogue	RoBERTa + DINO ViT + RankSum-BART	0.7902	0.7994	0.7422
	MPNet + DINO ViT + RankSum-BART	0.8475	0.8568	0.7550
	RoBERTa + ResNet + RankSum-BART	0.8837	0.8921	0.8259
	MPNet + ResNet + RankSum-BART	**0.9104**	**0.9036**	**0.8665**

Table 1: Evaluation of baseline models and multi-modal models on the fake news detection task, reporting the accuracy values for 2-, 3-, and 6-way classification. Note that the models refer to the text encoder model, the image encoder model, and the dialogue summarization model, respectively.

8.1.1 Impact of Dialogue Embeddings with RankSum

We find that the incorporation of dialogue data through ranked dialogue summarization into our multi-modal models does indeed improve performance, as recorded in Table 2. When comparing within the MPNet–ResNet architecture, adding the dialogue modality increases absolute accuracy by 8.72%, 11.92%, and 13.77% for 2-, 3-, and 6-way fake news classification, respectively. (These are relative accuracy increases of 10.59%, 15.20%, and 18.89%, respectively.)

It is interesting to note that the finer-grained fake news detection tasks experienced larger performance improvements with the incorporation of the dialogue modality; in other words, the 6-way fake news classification task had the largest performance improvement when using a multi-modal model

Measure	Models	Performance Improvements		
		2-way	3-way	6-way
Absolute Acc.	RoBERTa + DINO ViT + RankSum-BART	+6.89%	+10.22%	+5.51%
	MPNet + DINO ViT + RankSum-BART	+5.04%	+10.09%	+5.58%
	RoBERTa + ResNet + RankSum-BART	+7.50%	+11.05%	+11.88%
	MPNet + ResNet + RankSum-BART	**+8.72%**	**+11.92%**	**+13.77%**
Relative Acc.	RoBERTa + DINO ViT + RankSum-BART	+9.55%	+14.66%	+8.02%
	MPNet + DINO ViT + RankSum-BART	+6.32%	+13.35%	+7.98%
	RoBERTa + ResNet + RankSum-BART	+9.27%	+14.14%	+16.80%
	MPNet + ResNet + RankSum-BART	**+10.59%**	**+15.20%**	**+18.89%**

Table 2: Performance improvements measured in terms of absolute accuracy and relative accuracy when using multi-modal text–image–dialogue models (i.e., by including dialogue data via dialogue summarization), when compared to the baseline of using text–image models.

that incorporated dialogue data via ranked dialogue summarization. (On the other hand, 2-way fake news classification had the least amount of performance improvement.) We hypothesize that this is because the image and text modalities may be "enough" to classify a post as misinformation in a binary fashion, but the dialogue modality may provide additional information to help distinguish in a fine-grained manner which type of fake news it is (i.e., satire, false connection, imposter content, manipulated content, and misleading content, as defined in 3.1).

8.1.2 Comparison with Raw Dialogue Embeddings

It is also interesting to note that incorporating dialogue data could have been a potential source of noise for the overall model, particularly since user dialogue online (e.g., comment threads, tweet replies) is often not highly-moderated and could essentially contain any text. The RankSum approach leverages utterance metadata to produce a filtered, "cleaner" representation of the dialogue containing the most salient information; we verify that this occurs by comparing it against simply embedding the raw dialogue with a text encoder, with no ranking or filtering process. Our results are recorded in Table 3.

Models (with Dialogue Method)		Results		
		2-way	3-way	6-way
MPNet + ResNet (no dialogue)		0.8232	0.7844	0.7288
MPNet + ResNet + RawDialogue-MPNet*		0.8002	0.8094	0.7322
MPNet + ResNet + RankSum-BART		**0.9104**	**0.9036**	**0.8665**
RankSum*'s performance improvement over raw dialogue embeddings*	Absolute Accuracy	+11.02%	+9.42%	+13.43%
	Relative Accuracy	+13.77%	+11.64%	+18.34%

Table 3: Exploring the benefit of generating a summary of the dialogue, rather than using the dialogue utterances as-is. We also provide the text–image modality results again as a control, illustrating that the raw dialogue introduces noise for 2-way classification. **Note*: The raw dialogue is simply encoded with the MPNet text encoder.

As shown, RankSum performs better than simply incorporating the raw dialogue (and the raw dialogue actually hinders performance in the 2-way classification setting). We hypothesize that our

method of sorting the comments by upvotes (and more generally, by using metadata available with individual dialogue utterances) and truncating comments which exceeded the BART Transformer summarization pipeline's input token length helped to both emphasize the high-value comments as important (based on their location at the beginning of the corpus prior to summarization) and remove low-value comments (in the case in which there were many comments and thus truncating was required).

8.1.3 Impact of Text and Image Encoders

We also briefly note the impact of our various text and image encoders. When comparing the text encoder models of RoBERTa and MPNet, we find that MPNet outperforms RoBERTa in all experiment configurations, and adopt it in following experiment suites.

When comparing the performance of DINO ViT with ResNet for embedding images, we find that DINO ViT did *not* outperform ResNet, and and at times even hindered performance, disproving our initial hypothesis about leveraging Vision Transformers for better image embeddings. This could likely be due to the fact that the images did not benefit from the additional semantic segmentation that DINO ViT is able to provide, and as a result, the larger number of parameters associated with DINO ViT overparameterized the model. Given this and the additional model complexity of using ViTs, ResNet remains the better choice; we adopt it in the following experiment suites.

8.2 ArgSum: Evaluation of Summary Quality

Our next suite of experiments involved training and running hyperparameter tuning for ArgSum's submodels (for argumentative unit classification and entailment-based relationship type classification). Afterwards, we evaluated ArgSum's summary quality in general, before applying it in our multi-modal models.

8.2.1 Comparing ROUGE for ArgSum-BART and Baseline BART

We evaluated the summary quality of ArgSum-BART (i.e., constructing an argument graph from dialogue utterances, linearizing the argument graph, and then summarizing it with a BART Transformer pipeline) with that of a BART baseline (i.e., concatenating the dialogue utterances as-is and then summarizing with a BART Transformer pipeline), using the ROUGE metric on the test set of the SAMSum conversation summarization dataset. The ROUGE scores for both ArgSum-BART and baseline BART are recorded in Table 4, showing the slight improvements of ArgSum-BART.

Method	**Results**		
	ROUGE-1	**ROUGE-2**	**ROUGE-L**
Baseline BART	31.65	11.93	28.32
ArgSum-BART	32.27	13.12	28.46

Table 4: Comparing the summarization quality of baseline BART with ArgSum-BART on the test set of the SAMSum dataset for conversation summarization, after fine-tuning.

It is important to note, however, that these ROUGE score improvements are not huge; this, as was similarly found in Fabbri et al. (2021), may be because the SAMSum dataset's conversations do not benefit as much from modeling argumentative structures as other types of data (like comment threads); thus, future work benchmarking ROUGE scores for ArgSum on additional types of conversation data may be beneficial.

8.3 Multi-Modal Models with ArgSum: Evaluation on Detection Tasks

8.3.1 Comparing RankSum, GraphLin, and ArgSum for the Dialogue Modality

Our next suite of experiments incorporated ArgSum and GraphLin (i.e., running ArgSum but stopping after linearizing the graph and directly embedding that as the dialogue modality representation) into our multi-modal models for fake news detection. Our results are recorded in Table 5.

Modality	Models	Results		
		2-way	**3-way**	**6-way**
Text + Image + Dialogue	MPNet + ResNet + RankSum-BART	0.9104	0.9036	0.8665
	MPNet + ResNet + GraphLin-MPNet*	0.9125	**0.9326**	0.9116
	MPNet + ResNet + ArgSum-BART	**0.9208**	0.9216	**0.9172**

Table 5: Evaluation of text–image–dialogue multi-modal models using the three dialogue methods of RankSum, GraphLin, and ArgSum on the fake news detection task, reporting the accuracy values for 2-, 3-, and 6-way classification. **Note*: GraphLin uses the linearized argument graph directly as input without summarization, and thus it does not use the BART summarization pipeline, but rather encodes the linearized text directly with the MPNet text encoder.

We find that modeling dialogues through their argumentative structures provides performance gains over ranked dialogue summarization. There is no clear "best" method, however, for incorporating the argumentative structures; the linearized argument graphs via GraphLin and the argument graph–based summary via ArgSum perform similarly well over RankSum, and no one method outperforms the other in all k-way classification tasks. Nonetheless, this supports the notion that modeling argumentative structures for social media dialogues is a powerful approach in content moderation tasks.

8.3.2 Exploring the Salience Filtering Ability of ArgSum

We also note that ArgSum and GraphLin both outperformed the raw dialogue embeddings, as was seen in Table 3. This illustrates how the ArgSum algorithm is able to both group salient major claims and the premises which support them into continuous sequences (via the graph linearization process) and filter out non-argumentative units and place less emphasis on premises which do not entail any claims (via the linearization heuristics), the latter of which is similar to the ranked dialogue summarization, except that ArgSum does *not* require any utterance metadata (such as upvotes).

8.4 Comparison of Multi-Modal Fusion Methods

Our final suite of experiments aimed to explore better modeling of cross-modal interactions through low-rank tensor fusion. For all prior experiments, we used uni-modal early fusion (described in 6.1); we then took the best-performing models and incorporated low-rank tensor fusion (described in 6.2).

8.4.1 Evaluating Fusion Methods for Fake News Detection

Our results for comparing uni-modal early fusion and low-rank tensor fusion on the fake news detection task are recorded in Table 6.

We find that across all model configurations and modalities in this suite of experiments, low-rank tensor fusion outperforms uni-modal early fusion. The largest performance increase was in bi-modal 3-way classification; given that the bi-modal tensor fusion model did not reduce the dimensionality of the uni-modal tensors prior to fusion (since they had a computationally-feasible number of

Modality	Models	Uni-Modal Early Fusion			Low-Rank Tensor Fusion		
		2-way	3-way	6-way	2-way	3-way	6-way
Text + Image	MPNet + ResNet	0.8232	0.7844	0.7288	0.8325	0.8328	0.7473
Text + Image + Dialogue	MPNet + ResNet + RankSum-BART	0.9104	0.9036	0.8665	0.9248	0.9238	0.8988
	MPNet + ResNet + GraphLin-BART	0.9208	0.9216	0.9172	0.9301	0.9475	0.9227
	MPNet + ResNet + ArgSum-BART	0.9125	0.9326	0.9116	0.9312	0.9409	0.9298

Table 6: Comparing multi-modal fusion methods via evaluation of multi-modal models on the fake news detection task, reporting the accuracy values for 2-, 3-, and 6-way classification.

parameters without reduction), we hypothesize that this allows the cross-modal interactions to be modeled fully between the text and image modalities. However, the purpose of *low-rank* fusion is to be scalable to multiple modalities, and even when applying the tensor dimension reduction in the tri-modal settings, low-rank tensor fusion still outperformed uni-modal early fusion, illustrating the benefits of modeling cross-modal interactions in multi-modal models.

8.4.2 Evaluating Fusion Methods for Hate Speech Detection

As discussed in 3.2, the Twitter API did not allow us to scrape reply tweets to the original tweets in the MMHS150K dataset, since the original tweets had been flagged and taken down as hate speech since being collected in real-time by Gomez et al. (2019). Although MMHS150K does not have dialogue data to model with ArgSum as a result of this, it does still have three data modalities, since it contains OCR data (i.e., text within the image that was extracted via Optical Character Recognition), making it a strong candidate for experimenting with fusion methods. Our results for comparing uni-modal early fusion and low-rank tensor fusion on the hate speech detection task are recorded in Table 7.

Modality	Models	Uni-Modal Early Fusion	Low-Rank Tensor Fusion
Text + Image + OCR	MPNet + ResNet + MPNet	0.8653	0.8937

Table 7: Comparing multi-modal fusion methods via evaluation on the hate speech detection task, reporting the accuracy values for 6-way classification. Recall that OCR refers to text within an image, which was extracted using Optical Character Recognition and is encoded with the MPNet text encoder.

We again find that low-rank tensor fusion again outperformed uni-modal early fusion. We posit that this is particularly helpful for the hate speech detection task, since as Gomez et al. (2019) exemplify in their original paper, often times an individual text and an individual image are not hateful in and of themselves, but when posted together are a form of hate speech; incorporating cross-modal interactions through the differentiable outer product in low-rank fusion allows for this phenomenon to be adequately modeled.

9 Discussion and Future Work

9.1 Impact of Leveraging the Dialogue Modality for Content Moderation

Our experiments illustrated the power of leveraging the dialogue modality in multi-modal models for content moderation tasks, as quantified by the performance improvements of the text–image–dialogue models over their text–image and uni-modal baselines. Given that raw dialogue can also act as a source of noise and spam, particularly in less-moderated comment threads and reply systems on social media platforms, this does not simply mean adding more dialogue data will benefit the system; as seen in the raw dialogue embedding experiments, this could even hinder performance. Thus, effectively modeling the dialogue modality is critical, and the results of our experiments show that both ranked dialogue summarization leveraging utterance metadata (via RankSum) and modeling argumentative structures as linearized argument graphs (via GraphLin) and through argument graph–based summarization (via ArgSum) are effective methods for incorporating dialogue data into multi-modal models.

Future work can continue to experiment with methods of modeling the dialogue modality on additional types of social media data for content moderation tasks. Given the promising results of modeling argumentative structures within dialogues in general, future work can also experiment with applying these approaches on other downstream tasks.

9.2 Application of Multi-Modal Models to Social Media Platforms

The multi-modal models developed in this thesis were designed and implemented with scalability for production in mind. By developing all models with PyTorch and PyTorch Lightning, we encourage future work to build on our models and continue to develop models which are able to be easily trained and have their trained model assets be used in production-level systems. It would be interesting to see how these systems (and other similar multi-modal systems) could be applied to real-time content moderation on social media platforms.

9.3 Extensibility of `argsum` Software Package for Modeling Dialogues

The `argsum` software package in our codebase, which implements the ArgSum algorithm, can be applied for modeling dialogues in general. It provides classes for the argument graph data structures (in `argsum.data_structures`); module classes for the various steps of the ArgSum algorithm, i.e., utterance segmentation, graph construction, and graph linearization (in `argsum.modules`); and PyTorch and PyTorch Lightning models for the argument graph construction submodels (in `argsum.submodels`). This can be applied to model dialogues as argument graphs for not only dialogue summarization, but other downstream tasks as well. Additionally, future work could explore using graph neural networks and graph encoders to directly embed the argument graphs produced by the argument graph construction step of ArgSum.

10 Conclusion

Multi-modal models offer a powerful approach in detecting misinformation and hate speech online, particularly on social media platforms where user dialogue can be incorporated as an additional modality, giving more insight into the integrity or hatefulness of a post. This thesis developed multi-modal models for the content moderation tasks of fake news detection and hate speech detection, finding that the incorporation of the dialogue modality through ranked dialogue summarization, linearized argument graphs, and argument graph–based dialogue summarization was able to improve performance. Additionally, we introduced ArgSum as a software package for the modeling of argumentative structures in dialogues and evaluated its general summarization quality. Finally, we found that low-rank tensor fusion with the computation of differentiable outer products was able to further improve the modeling of cross-modal interactions better than early fusion methods. We hope that our work can contribute to the research fields of multi-modal machine learning, dialogue modeling and summarization, and content moderation, and inspire future work for the reduction of misinformation and hate speech online.

Acknowledgements

I would like to thank Dr. Dragomir Radev for his mentorship during my time at Yale, from when I first joined his Language, Information, and Learning (LILY) research lab my sophomore year, throughout his CS 477 and CS 677 courses my junior and senior years, and through the development of this senior project. The lab sparked my interest in natural language processing and gave me exposure to working on dialogue summarization through the ConvoSumm project, and also provided a strong foundation for working on NLP and applied machine learning in internships later on. I look forward to applying the skills I learned from Drago in my future career, and am grateful to have studied under him.

Appendix A Overview of Code Repository

Our codebase is available at https://github.com/faiazrahman/Multimodal-Content-Moderation.

- `environment.yaml`: Conda environment file to recreate our virtual environment from

- `data_preprocessing.py`: Script to run data preprocessing for Fakeddit
- `dataloader.py`: Contains the `torch.utils.data.Dataset` for fake news detection
- `data/`: Contains a subdirectory for each of the five datasets with Bash scripts and/or instructions to download each (and additional subdirectories for the aggregated argumentative unit classification and aggregated relationship type classification data)
- `data_sampling/`: Contains scripts to generate samples of the Fakeddit dataset with an even class distribution

- `run_training.py`: Runs training for fake news detection models
- `run_evaluation.py`: Runs evaluation for fake news detection models
- `run_experiments.py`: Automates multiple training and evaluation runs via Python's `subprocess` module (i.e., running the shell commands simultaneously, similar to a regular Bash script except it also generates log files with specific filename conventions based on the experiment configs)
- `utils.py`: Various utilities for running experiments and evaluation (e.g. getting the latest model checkpoint from the trained assets folder, etc.)

- `models/`
  - Contains all multi-modal models and uni-modal baseline models for fake news detection
    - ⇒ Each file in this subdirectory contains a `pl.LightningModule` (which is the model that is imported during training and evaluation) and a PyTorch `nn.Module` (which implements the model architecture and is used internally by the Lightning model)

- `argsum/` (formerly `argument_graphs/`)
  - `argsum/`
    - ⇒ Contains the `ArgSum` class, which can be instantiated to generate argument graphs, linearized argument graphs, and dialogue summaries via the ArgSum algorithm
  - `data_structures/`
    - ⇒ Contains the data structures for working with argument graphs; specifically, the classes `ArgumentGraph`, `ArgumentativeUnitNode`, `RelationshipTypeEdge`, `ArgumentativeUnitType`, and `RelationshipType`
  - `modules/`
    - ⇒ Contains the modules for various steps of the ArgSum algorithm; specifically, `utterance_segmentation.py`, `graph_construction.py`, and `graph_linearization.py`, which contain the `UtteranceToArgumentativeUnitSegmenter`, `ArgumentGraphConstructor`, and `ArgumentGraphLinearizer` classes, respectively
  - `submodels/`
    - ⇒ Contains the models used during argument graph construction; specifically, `ArgumentativeUnitClassificationModel` and `RelationshipTypeClassification`, which are both `pl.LightningModule`s with an underlying PyTorch `nn.Module`
  - `utils/`
    - ⇒ Contains various utilities for the tokenizers used by ArgSum's submodels and for generating batches during inference

- `data_preprocessing.py`: Runs data preprocessing for the AMPERSAND, Stab & Gurevych, and MNLI datasets
- `dataloader.py`: Contains the `torch.utils.data.Dataset`s for the ArgSum submodels
- `run_argument_graph_submodel_training.py`: Runs training for the ArgSum submodels
- `run_argument_graph_submodel_evaluation.py`: Runs evaluation for the ArgSum submodels
- `run_hyperparameter_tuning.py`: Runs hyperparameter tuning for the ArgSum submodels
- `run_hyperparameter_tuning_evaluation.py`: Runs evaluation for ArgSum submodel hyperparameter trials

- `dialogue_summarization/`
  - Contains data preprocessing for SAMSum, and ROUGE evaluation scripts for ArgSum-BART and baseline BART

- `fusion/`
  - Contains data preprocessing, the `torch.utils.data.Dataset`, the model, the train and evaluation scripts, and an automated experiments script for multi-modal hate speech detection, which is used for fusion method experiments
    - ⇒ Note that these same fusion methods are also implemented in the multi-modal fake news detection models in the root's `models/`

- `configs/`
  - Contains all experiment configurations as YAML files, which are passed to training and evaluation scripts via the `--config` arg

- `lightning_logs/`
  - This is where all trained model assets (i.e., checkpoints and hparams) are stored by PyTorch Lightning

- `logs/`
  - This is where the logs generated by `run_experiments.py` (and the other automated run scripts in the repo) are stored

Appendix B Algorithms

We provide the pseudocode for the ArgSum algorithm (which itself consists of the graph construction and GraphLin algorithms) in the following. Their implementations can be found in our codebase.

Algorithm 1 Argument graph construction for ArgSum

Input: D, the set of dialogue utterances
Output: $G = (V, E)$, an argument graph

let $U \leftarrow$ `[]` ▷ Utterance segmentation
for $d \in D$ **do**
  $u \leftarrow$ `UtteranceToArgumentativeUnitSegmenter(`d`)`
  $U \leftarrow$ `[`$...U, u$`]`
end for

Load trained AUC model and its tokenizer ▷ Argumentative unit classification
for batch of $u \in U$ **do**
  Pass batch of argumentative units through AUC model to get classification predictions
  Create an `ArgumentativeUnitNode` *for each argumentative unit*
end for

let $G \leftarrow$ *Instantiate an* `ArgumentGraph` *object* ▷ Relationship type classification

Load trained RTC model and its tokenizer ▷ (a) premise-to-claim entailment
for $p \in$ all premise nodes $P \subseteq U$ **do**
  Compute the entailment score for this premise against all claims (passing in batches to the RTC model to get the relationship type predictions) and get the claim c_{max} with the maximum score
  if the maximum score is above the minimum entailment score threshold **then**
    Create a support `RelationshipTypeEdge` *from the premise p to the claim c_{max}*
  end if
end for

First, store all potential claim-to-claim edges ▷ (b) claim-to-claim entailment
for $c \in$ all claims $C \subseteq U$ **do**
  let $C' \leftarrow C \setminus c$
  Compute the entailment score for this claim c against all other claims $c' \in C'$ (again passing in batches to the RTC model to get the relationship type predictions) and get the claim c'_{max} with the maximum score
  if the maximum score is above the minimum entailment score threshold **then**
    Store a potential support edge from c to c'_{max} (but do not add it to G yet)
  end if
end for

Next, greedily add edges from the stored potential edges in order of decreasing entailment score, only if it does not create a cycle in the graph

let root node $r \leftarrow$ `ArgumentativeUnitNode` with classification `ROOT` ▷ Root node linking
for $u \in U$ **do**
  if the node u does not entail any other nodes **then**
    Create an edge from the node to the root, with relationship type `TO_ROOT`
  end if
end for

Algorithm 2 GraphLin: Heuristic-based argument graph linearization

Input: $G = (V, E)$, an argument graph consisting of a set V of argumentative unit nodes and a set E of relationship type edges between them
Output: The linearized argument graph, as a string

let $L \leftarrow$ `[]`

Pre-compute claim subtree sizes in G

let stack s, set of visited nodes V
Add the child claims of the root to the stack in order of subtree size, s.t. the claim with the largest subtree is at the top (and will be processed first)

while stack s is not empty **do**
  let current claim node $c \leftarrow s$`.pop()`
  if $c \in V$ **then**
    continue (to the next iteration of the loop)
  end if
  Add c to V

  $L \leftarrow$ `[`$...L, c$`]` ▷ Greedy Step: Add the current claim node immediately

  Add child premises of c to L ▷ Semantic Ordering Heuristic
  (i.e., child premises are added before child claims)

  Add child claims of c to L in order of smallest ▷ Subtree Size Prioritization Heuristic
  subtree to largest (so that in our next iteration, we pop the largest child subtree), i.e. large subtrees before small ones
end while

After traversal, add child premises of the root ▷ Zero-Degree Premise Tailing Heuristic
to the linearized string, i.e., at the very end

Stringify the linearized graph L and return

Algorithm 3 ArgSum: Argument graph–based dialogue summarization

Input: D, the set of dialogue utterances
Output: The summary of D

1. Construct an argument graph G from running Algorithm 1 on D.
2. Linearize the argument graph by running GraphLin, i.e., Algorithm 2, on G, producing the linearized string s.
3. Run s through a sequence-to-sequence Transformer summarization pipeline to produce a summary. (In our implementation, we use BART.)

Bibliography

Hunt Allcott, Matthew Gentzkow, and Chuan Yu. 2019. Trends in the diffusion of misinformation on social media. *Research & Politics*, 6(2):205316801984855.

Tadas Baltrusaitis, Chaitanya Ahuja, and Louis-Philippe Morency. 2017. Multimodal machine learning: A survey and taxonomy. *CoRR*, abs/1705.09406.

Emma Barker and Robert Gaizauskas. 2016. Summarizing multi-party argumentative conversations in reader comment on news. In *Proceedings of the Third Workshop on Argument Mining (ArgMining2016)*, pages 12–20, Berlin, Germany. Association for Computational Linguistics.

Filip Boltužić and Jan Šnajder. 2020. Structured prediction models for argumentative claim parsing from text. *Automatika*, 61(3):361–370.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. 2021. Emerging properties in self-supervised vision transformers. *CoRR*, abs/2104.14294.

Tommaso Caselli, Valerio Basile, Jelena Mitrović, and Michael Granitzer. 2021. HateBERT: Retraining bert for abusive language detection in english.

Tuhin Chakrabarty, Christopher Hidey, Smaranda Muresan, Kathy McKeown, and Alyssa Hwang. 2019. AMPERSAND: Argument mining for PERSuAsive oNline discussions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2933–2943, Hong Kong, China. Association for Computational Linguistics.

Johannes Daxenberger, Benjamin Schiller, Chris Stahlhut, Erik Kaiser, and Iryna Gurevych. 2020. ArgumenText: Argument classification and clustering in a generalized search scenario. *Datenbank-Spektrum*, 20(2):115–121.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Daisy Yi Ding and Robert Tibshirani. 2021. Cooperative learning for multi-view analysis. *arXiv preprint arXiv:2112.12337*.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929.

Alexander Fabbri, Faiaz Rahman, Imad Rizvi, Borui Wang, Haoran Li, Yashar Mehdad, and Dragomir Radev. 2021. ConvoSumm: Conversation summarization benchmark and improved abstractive summarization with argument mining. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6866–6880, Online. Association for Computational Linguistics.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. 2019. SAMSum corpus: A human-annotated dialogue dataset for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 70–79, Hong Kong, China. Association for Computational Linguistics.

Raul Gomez, Jaume Gibert, Lluis Gomez, and Dimosthenis Karatzas. 2019. Exploring hate speech detection in multimodal publications.

Wenzhong Guo, Jianwen Wang, and Shiping Wang. 2019. Deep multimodal representation learning: A survey. *IEEE Access*, 7:63373–63394.

Casper Hansen, Christian Hansen, and Lucas Chaves Lima. 2021. Automatic fake news detection: Are models learning to reason?

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition.

Jack Hessel and Lillian Lee. 2020. Does my multimodal model learn cross-modal interactions? it's harder to tell than you might think! In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 861–877, Online. Association for Computational Linguistics.

Jessica Katz Jameson and Nicole M. Lee. 2020. Introduction to the special issue on dialogue 2.0: New perspectives, enduring challenges, and promising directions. *Social Media Society*, 6(4):205630512098446.

Siddhant M. Jayakumar, Wojciech M. Czarnecki, Jacob Menick, Jonathan Schwarz, Jack W. Rae, Simon Osindero, Yee Whye Teh, Tim Harley, and Razvan Pascanu. 2020. Multiplicative interactions and where to find them. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Hamid Karimi, Proteek Roy, Sari Saba-Sadiya, and Jiliang Tang. 2018. Multi-source multi-class fake news detection. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1546–1557, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Douwe Kiela, Hamed Firooz, Aravind Mohan, Vedanuj Goswami, Amanpreet Singh, Pratik Ringshia, and Davide Testuggine. 2021. The hateful memes challenge: Detecting hate speech in multimodal memes.

Mirko Lenz, Premtim Sahitaj, Sean Kallenberg, Christopher Coors, Lorik Dumani, Ralf Schenkel, and Ralph Bergmann. 2020. Towards an argument mining pipeline transforming texts to argument graphs. *arXiv preprint arXiv:2006.04562*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. 2019. VisualBERT: A simple and performant baseline for vision and language.

Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, Yejin Choi, and Jianfeng Gao. 2020. Oscar: Object-semantics aligned pre-training for vision-language tasks.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Sander van der Linden, Jon Roozenbeek, and Josh Compton. 2020. Inoculating against fake news about COVID-19. *Frontiers in Psychology*, 11.

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.

Zhun Liu, Ying Shen, Varun Bharadhwaj Lakshminarasimhan, Paul Pu Liang, AmirAli Bagher Zadeh, and Louis-Philippe Morency. 2018. Efficient low-rank multimodal fusion with modality-specific factors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2247–2256, Melbourne, Australia. Association for Computational Linguistics.

Mengmeng Ma, Jian Ren, Long Zhao, Sergey Tulyakov, Cathy Wu, and Xi Peng. 2021. SMIL: multimodal learning with severely missing modality. *CoRR*, abs/2103.05677.

Ariadna Matamoros-Fernández and Johan Farkas. 2021. Racism, hate speech, and social media: A systematic review and critique. *Television & New Media*, 22(2):205–224.

Binny Mathew, Ritam Dutt, Pawan Goyal, and Animesh Mukherjee. 2019. Spread of hate speech in online social media. In *Proceedings of the 10th ACM Conference on Web Science*. ACM.

Kai Nakamura, Sharon Levy, and William Yang Wang. 2020. r/Fakeddit: A new multimodal benchmark dataset for fine-grained fake news detection.

Frederik Pahde, Mihai Marian Puscas, Tassilo Klein, and Moin Nabi. 2020. Multimodal prototypical networks for few-shot learning. *CoRR*, abs/2011.08899.

Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. 2018. Automatic detection of fake news. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3391–3401, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Hai Pham, Paul Pu Liang, Thomas Manzini, Louis-Philippe Morency, and Barnabás Póczos. 2018. Found in translation: Learning robust joint representations by cyclic translations between modalities. *CoRR*, abs/1812.07809.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Anil Rahate, Rahee Walambe, Sheela Ramanna, and Ketan Kotecha. 2022. Multimodal co-learning: Challenges, applications with datasets, recent advances and future directions. *Information Fusion*, 81:203–239.

Kai Shu, Deepak Mahudeswaran, Suhang Wang, Dongwon Lee, and Huan Liu. 2019. FakeNewsNet: A data repository with news content, social context and spatialtemporal information for studying fake news on social media.

Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake news detection on social media: A data mining perspective. *SIGKDD Explor. Newsl.*, 19(1):22–36.

Amanpreet Singh, Ronghang Hu, Vedanuj Goswami, Guillaume Couairon, Wojciech Galuba, Marcus Rohrbach, and Douwe Kiela. 2021. FLAVA: A foundational language and vision alignment model. *CoRR*, abs/2112.04482.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. MPNet: Masked and permuted pre-training for language understanding.

Christian Stab and Iryna Gurevych. 2014. Identifying argumentative discourse structures in persuasive essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 46–56, Doha, Qatar. Association for Computational Linguistics.

Jabeen Summaira, Xi Li, Amin Muhammad Shoib, Songyuan Li, and Jabbar Abdul. 2021. Recent advances and trends in multimodal deep learning: A review.

Shardul Suryawanshi, Bharathi Raja Chakravarthi, Mihael Arcan, and Paul Buitelaar. 2020. Multimodal meme dataset (MultiOFF) for identifying offensive content in image and text. In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*, pages 32–41, Marseille, France. European Language Resources Association (ELRA).

Shalini Talwar, Amandeep Dhir, Dilraj Singh, Gurnam Singh Virk, and Jari Salo. 2020. Sharing of fake news on social media: Application of the honeycomb framework and the third-person effect hypothesis. *Journal of Retailing and Consumer Services*, 57:102197.

Hao Tan and Mohit Bansal. 2020. Vokenization: Improving language understanding with contextualized, visual-grounded supervision. *CoRR*, abs/2010.06775.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: a large-scale dataset for fact extraction and verification.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

N. Velásquez, R. Leahy, N. Johnson Restrepo, Y. Lupu, R. Sear, N. Gabriel, O. K. Jha, B. Goldberg, and N. F. Johnson. 2021. Online hate network spreads malicious COVID-19 content outside the control of individual social media platforms. *Scientific Reports*, 11(1).

Prashanth Vijayaraghavan, Hugo Larochelle, and Deb Roy. 2021. Interpretable multi-modal hate speech detection.

Soroush Vosoughi, Deb Roy, and Sinan Aral. 2018. The spread of true and false news online. *Science*, 359(6380):1146–1151.

Weiyao Wang, Du Tran, and Matt Feiszli. 2019. What makes training multi-modal networks hard? *CoRR*, abs/1905.12681.

William Yang Wang. 2017. "Liar, Liar Pants on Fire": A new benchmark dataset for fake news detection.

Yaqing Wang, Fenglong Ma, Haoyu Wang, Kishlay Jha, and Jing Gao. 2021. Multimodal emergent fake news detection via meta neural process networks. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery Data Mining*.

Thilini Wijesiriwardene, Hale Inan, Ugur Kursuncu, Manas Gaur, Valerie L. Shalin, Krishnaprasad Thirunarayan, Amit Sheth, and I. Budak Arpinar. 2020. ALONE: A dataset for toxic behavior among adolescents on twitter.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Amir Zadeh, Minghai Chen, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2017. Tensor fusion network for multimodal sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1103–1114, Copenhagen, Denmark. Association for Computational Linguistics.

Chao Zhang, Zichao Yang, Xiaodong He, and Li Deng. 2020. Multimodal intelligence: Representation learning, information fusion, and applications. *IEEE Journal of Selected Topics in Signal Processing*, 14(3):478–493.

Pengchuan Zhang, Xiujun Li, Xiaowei Hu, Jianwei Yang, Lei Zhang, Lijuan Wang, Yejin Choi, and Jianfeng Gao. 2021. VinVL: Revisiting visual representations in vision-language models.