MAMKit: A Comprehensive Multimodal Argument Mining Toolkit

Eleonora Mancini and **Federico Ruggeri** and **Stefano Colamonaco**
Andrea Zecca and **Samuele Marro** and **Paolo Torroni**
DISI, University of Bologna, Bologna, Italy
{e.mancini, federico.ruggeri6}@unibo.it

Abstract

Multimodal Argument Mining (MAM) is a recent area of research aiming to extend argument analysis and improve discourse understanding by incorporating multiple modalities. Initial results confirm the importance of paralinguistic cues in this field. However, the research community still lacks a comprehensive platform where results can be easily reproduced, and methods and models can be stored, compared, and tested against a variety of benchmarks. To address these challenges, we propose MAMKit, an open, publicly available, PyTorch toolkit that consolidates datasets and models, providing a standardized platform for experimentation. MAMKit also includes some new baselines, designed to stimulate research on text and audio encoding and fusion for MAM tasks. Our initial results with MAMKit indicate that advancements in MAM require novel annotation processes to encompass auditory cues effectively.

1 Introduction

Recent studies in argumentation analysis highlight the importance of including paralinguistic features in argumentative discourse analysis across various domains, including advertisements, news coverage, and legal analytics (Kišiček, 2014; Groarke and Kišiček, 2018). Similar considerations have been made for fake news detection (Ivanov et al., 2023). For these reasons, Multimodal Argument Mining (MAM) recently emerged as an extension of Argument Mining, aiming to validate these propositions empirically and gain a more comprehensive understanding of argumentative discourse by integrating multiple modalities. MAM is a growing research field. The tasks addressed so far include argument detection, argument component classification, relation classification, and fallacy classification (Lippi and Torroni, 2016a; Mestre et al., 2021a; Mancini et al., 2022; Mestre et al., 2023; Mancini et al., 2024). However, despite these encouraging results, and similarly to what happens in other domains (Li et al., 2023; Helwe et al., 2022), the lack of standardized tools is hampering progress since MAM researchers struggle to access and evaluate models and datasets. For one thing, MAM resources are often hosted across various sites and repositories, each employing its own distinct methods for loading and reconstructing datasets and models. As a consequence, a fair model comparison may be problematic, which in turn limits the experimental evaluation of new models.

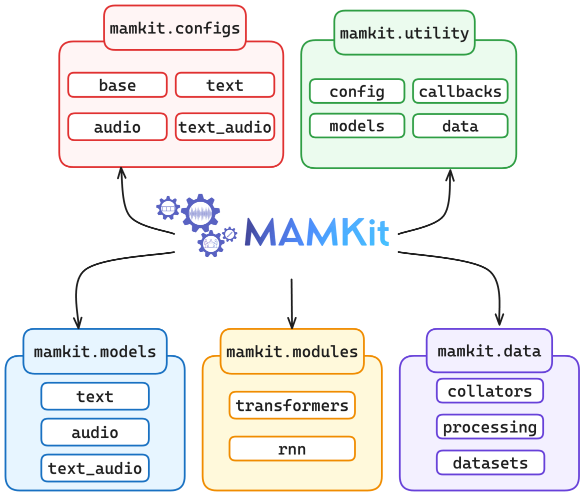

Figure 1: Overall architecture of MAMKit.

We then introduce a PyTorch toolkit tailored for MAM. Our toolkit, MAMKit, currently includes 4 datasets and 6 models, providing researchers with a standardized platform for experimentation and evaluation. MAMKit offers a simple interface to load, reconstruct and process existing datasets, and contribute new resources. Moreover, all models within MAMKit are implemented uniformly, facilitating seamless integration and comparison across tasks and datasets. To ensure the reproducibility and reliability of our models, all the resources and models in MAMKit have been validated against the

Proceedings of the 11th Workshop on Argument Mining, pages 69–82
August 15, 2024 ©2024 Association for Computational Linguistics

original research papers, offering a shared interface for benchmarking and model comparison. Besides literature models, MAMKit explores and integrates advanced audio encoding and fusion methods. Indeed, previous research in MAM has largely overlooked advanced audio encoding and fusion strategies (Mancini et al., 2024), thus MAMKit intends to present an opportunity to shed light on the significance of audio and the synergistic interaction between both modalities in argument mining tasks.

2 Related Work

We overview the MAM literature and the landscape of toolkits built to address relevant AI tasks in different application domains.

2.1 Multimodal Argument Mining

Work in MAM started relatively recently, inspired by studies on the connections between arguments and emotions (Benlamine et al., 2015), with the development of a classifier for claim detection from speech in the domain of political debates, and a small dataset built for the occasion (Lippi and Torroni, 2016a). The interest in political debates motivated further research and resource development (Lawrence and Reed, 2019; Mancini et al., 2022; Mestre et al., 2023; Mancini et al., 2024). Notably, Mancini et al. (2022) and Mestre et al. (2023) introduced two distinct expansions of USED (Haddadan et al., 2019), the US presidential election corpus. Recently, Mancini et al. (2024) proposed an extension of USED-fallacy, releasing the first corpus for multimodal fallacy classification. These resources constitute the most extensive multimodal corpora for AM to date. Another domain of interest is fake news detection. There, Ivanov et al. (2023) observed enhanced classification performance across various tasks, such as the identification of check-worthy claims, through the adoption of a multimodal formulation.

The MAM systems adopted in literature so far uncovered the importance of tackling argumentative tasks from a multimodal standpoint, but they did not introduce significant architectural innovations. On the contrary, they mostly followed the standard practice of merging unimodal models using fusion techniques (Toto et al., 2021): see for instance (Mancini et al., 2022; Mestre et al., 2023; Mancini et al., 2024). However, recent advancements in Multimodal Deep Learning (MMDL) offer an opportunity for exploring new architectural solutions. Some of the new models introduced in MAMKit extend previous work (Mancini et al., 2022) with new MAM models based on state-of-the-art models for audio encoding and multimodal fusion techniques (Boulahia et al., 2021). These include Wav2Vec 2.0 (Baevski et al., 2020), HuBERT (Hsu et al., 2021) and WavLM (Chen et al., 2022) for audio encoding, as well as early, cross-modal (Tsai et al., 2019) and late fusion.

2.2 Toolkits

In recent years, there has been a growing emphasis on streamlining training, evaluation, and benchmarking processes across diverse domains of artificial intelligence (AI). Accordingly, new resources became available to address specific tasks and applications. Regarding benchmarking, LAVIS (Li et al., 2023), MMF (Singh et al., 2020), X-modaler (Li et al., 2021) and UniLM (uni, 2020) provide user-friendly interfaces for accessing datasets and for training/evaluating language-vision models.

Furthermore, several tools have been proposed for multimodalities. Notable examples are TorchMultimodal (tor, 2022) for accessing several state-of-the-art multimodal models, ViLMedic (Delbrouck et al., 2022) for vision and language in medical AI, pyannote.metrics (Bredin, 2017) and pyannote.audio (Bredin, 2023) for speaker diarization, and Muskits (Shi et al., 2022) for end-to-end music processing.

Moreover, several specialized NLP libraries and tools focus on specific tasks. They include LogiTorch (Helwe et al., 2022) for logical reasoning in natural language, TextBox 2.0 (Tang et al., 2022) for text generation using pre-trained language models, mahaNLP (Magdum et al., 2023) for Marathi NLP, DeepPavlov (Burtsev et al., 2018) for dialogue systems, TextAttack (Morris et al., 2020) and OpenAttack (Zeng et al., 2021) for adversarial attacks in NLP, LambdaKG (Xie et al., 2023) for knowledge graph embeddings, nerblackbox (Stollenwerk, 2023) for named entity recognition, NewsRecLib (Iana et al., 2023) for news recommendation, and NeuralQA (Dibia, 2020) for question answering cater to specific NLP tasks.

To the best of our knowledge, there are no such resources to support argument mining/MAM research, so MAMKit is the first toolkit in this area.

3 MAMKit

MAMKit is an open-source, publicly available[1] PyTorch toolkit designed to access and develop datasets, models, and benchmarks for MAM. It provides a flexible interface for accessing and integrating datasets, models, and preprocessing strategies through composition or custom definition. MAMKit is designed to be extendible, ensure replicability, and provide a shared interface as a common foundation for experimentation in the field. At the time of writing, MAMKit offers 4 datasets and 6 distinct model architectures, along with audio and text processing capabilities, organized in 5 main components (see Figure 1).

3.1 Description of toolkit components

Datasets The `mamkit.data` package covers dataset creation (`data.datasets`) and preprocessing (`data.preprocessing` and `data.collators`). The `data.datasets` module provides the `Loader` interface, a general-purpose wrapper for datasets, covering data downloading, task-specific data parsing, and data interfacing. Regarding the latter functionality, the module includes ad-hoc implementations for unimodal (`UnimodalDataset`) and multimodal (`MultimodalDataset`) data based on the PyTorch `Dataset` interface. The `data.processing` module provides the `Processor` interface for defining custom data processing and implements unimodal (`UnimodalProcessor`) and multimodal (`MultimodalProcessor`) processing steps. For instance, the `AudioTransformer` class implements transformer-based audio processing. Similarly to `data.processing`, the `data.collators` module is designed to address input processing at batch-level, in compliance with PyTorch `DataLoader` APIs. The module includes implementations for unimodal (`UnimodalCollator`) and multimodal (`MultimodalCollator`) input batches.

Models The `mamkit.models` package holds definitions for the supported models. It provides `models.audio`, `models.text` and `models.text_audio` modules. Each model implements the `torch.nn.Module` interface that can be extended to define the models for each input configuration.

Modules The `mamkit.modules` package handles the definition of the shared model layers such as `transformer_modules`.

[1] `https://github.com/lt-nlp-lab-unibo/mamkit`

Utility The `mamkit.utility` package contains classes and methods used by other modules. For example, the `utility.data` module contains methods for downloading data from web storages or GitHub repositories, while `utility.model` manages the overall training and evaluation lifecycles. Currently, the `MAMKitLightningModel` class and the `to_lightning_model()` method are used to wrap models as PyTorch Lightning (Falcon and The PyTorch Lightning team, 2019) models to leverage its functionalities for training and evaluation. Incorporating PyTorch Lightning in our toolkit streamlines training and evaluation with a simplified loop, standardized interface, reproducibility, performance optimizations, accelerator integration, logging capabilities, and extensive community support.

Configs The `mamkit.configs` package serves as a streamlined interface for accessing model configurations across three modalities: audio, text, and text-audio. At its core, the `config.base` module establishes two fundamental classes: `ConfigKey`, defining configuration keys, and `BaseConfig`, providing a base configuration structure. This architecture simplifies benchmarking efforts by enabling users to instantiate models via designated configuration keys. Consequently, leveraging models with exact parameter setups for benchmarking or further experimentation becomes straightforward, enhancing research reproducibility and efficiency within the toolkit.

3.2 Example Usage

MAMKit's design facilitates access to existing datasets and models and supports future development. In this section, we present several examples to illustrate common use cases.

3.2.1 Data Loading

An important feature of MAMKit is its unified and straightforward interface for data access. Several MAM datasets are included in MAMKit. Adding a new dataset to MAMKit requires defining a new subclass of `Loader`, extending it with the specific information needed to access and reconstruct the dataset. In the example that follows, a dataset is loaded using the `MMUSED` class from `mamkit.data.datasets`, which extends the `Loader` interface and implements specific functionalities for data loading and retrieval. Users can specify task and input mode (`text-only`, audio-only, or text-audio) when loading the data, with options to use default splits or load splits from previous works. The example uses splits from Mancini et al. (2022).

```
from mamkit.data.datasets import UKDebates,
↪  InputMode

loader = UKDebates(
          task_name='asd',
          input_mode=InputMode.TEXT_ONLY,
          base_data_path=base_data_path)


split_info =
↪  loader.get_splits('mancini-et-al-2022')
```

The get_splits method of the loader returns data splits in the form of a data.datasets.SplitInfo. The latter wraps split-specific data, each implementing Pytorch's Dataset interface and compliant to the specified input modality (i.e., text-only).

The Loader interface also allows users to integrate methods defining custom splits as follows:

```
from mamkit.data.datasets import SplitInfo

def custom_splits(self) -> List[SplitInfo]:
    train_df = self.data.iloc[:50]
    val_df = self.data.iloc[50:100]
    test_df = self.data.iloc[100:]
    fold_info =
    ↪  self.build_info_from_splits(train_df=...)
    return [fold_info]

loader.add_splits(method=custom_splits,
                  key='custom')

split_info = loader.get_splits('custom')
```

3.2.2 Modelling

MAMKit offers a simple method for defining custom models and leveraging models from the literature. Utilizing the same interface for both tasks aims to simplify access to existing models and establish new ones with a coherent structure. This will hopefully facilitate the spread of established models and encourage the development of new ones by maintaining consistency throughout the process. The example below illustrates that defining a custom model is straightforward. It entails creating the model within the models package, specifically by extending either the AudioOnlyModel, TextOnlyModel, or TextAudioModel classes in the models.audio, models.text, or models.text_audio modules, respectively, depending on the input modality handled by the model.

```
from mamkit.models.text import Transformer

model = Transformer(
          model_card='bert-base-uncased',
          dropout_rate=0.1, ...)
```

The following example demonstrates how to instantiate a model with a configuration found in the literature. This configuration is identified by a key, ConfigKey, containing all the defining information. The key is used to fetch the precise configuration of the model from the configs package. Subsequently, the model is retrieved from the models package and configured with the specific parameters outlined in the configuration.

```
from mamkit.configs.base import ConfigKey
from mamkit.configs.text import TransformerConfig
from mamkit.data.datasets import InputMode

config_key = ConfigKey(
              dataset='mmused',
              task_name='asd',
              input_mode=InputMode.TEXT_ONLY,
              tags={'mancini-et-al-2022'})

config = TransformerConfig.from_config(
                          key=config_key)

model = Transformer(
         model_card=config.model_card,
         dropout_rate=config.dropout_rate
         ...)
```

In both the described use cases, the model is then encapsulated into a Pytorch Lightning model, and training and evaluation are conducted by leveraging the methods provided by this wrapper.

```
from mamkit.utility.model import
↪  to_lighting_model
import lightning

model = to_lighting_model(model=model,
        num_classes=config.num_classes,
        loss_function=...,
        optimizer_class=...)

trainer = lightning.Trainer(max_epochs=100,
                            accelerator='gpu',
                            ...)
trainer.fit(model,
            train_dataloaders=train_dataloader,
            val_dataloaders=val_dataloader)
```

3.2.3 Benchmarking

The mamkit.configs package simplifies reproducing literature results in a structured manner. Upon loading the dataset, experiment-specific configurations can be easily retrieved via a configuration key. Specifically, unlike the examples reported in Section 3.2.2, where configurations refer just to a model implementation, in the below example, they encompass both data processing and model parameterization based on previous literature work.

This enables instantiating a processor using the same features processor employed in the experiment. In the example below, we adopt a configuration akin to (Mancini et al., 2022), employing a BiLSTM model with audio encoded with MFCCs features. Hence, we define a `MFCCExtractor` processor using configuration parameters. Data splits are loaded using the experiment reference key, mirroring what was shown in Section 3.2.1.

```
from mamkit.configs.audio import
↪ BiLSTMMFCCsConfig
from mamkit.configs.base import ConfigKey
from mamkit.data.datasets import UKDebates,
↪ InputMode
from mamkit.data.processing import MFCCExtractor,
↪ UnimodalProcessor
from mamkit.models.audio import BiLSTM

loader = UKDebates(task_name='asd',
           input_mode=InputMode.AUDIO_ONLY)

config = BiLSTMMFCCsConfig.from_config(
              key=ConfigKey(dataset='ukdebates',
                input_mode=InputMode.AUDIO_ONLY,
                task_name='asd',
                tags='mancini-et-al-2022'))


for split_info in loader.get_splits(
                     key='mancini-et-al-2022'):
    processor =
        UnimodalProcessor(
            features_processor=MFCCExtractor(
                mfccs=config.mfccs, ...))

    split_info.train =
    ↪   processor(split_info.train)
    ...
    model = BiLSTM(embedding_dim=
                config.embedding_dim, ...)
```

3.3 Models

MAMKit comes with 3 models from the MAM literature and 3 original models we contribute based on state-of-the-art unimodal audio encoders and fusion strategies. All models comply with the following architecture: text and audio modules for encoding individual modalities, a fusion layer to merge them, and a final classification head tailored to the downstream task of interest. Table 1 provides a summary. Illustrations of our original architectures are shown in Appendix A. We refer to the fusion strategies as follows:

- **Concatenation**: combines features (*early fusion*) or embeddings from single modality architectures (*late fusion*) from all modalities into a single vector by concatenating them;
- **Average**: merges features (*early fusion*) or embeddings from single modality architectures (*late fusion*) by simply averaging information from each modality;
- **Crossmodal Attention**: attends to interactions between multimodal sequences across distinct time steps and facilitates the transfer of streams from one modality to another.

BiLSTM (Mancini et al., 2022) The text module comprises a pre-trained GloVe (Pennington et al., 2014) embedding layer and a stack of BiLSTM layers. Similarly, the audio module is a stack of BiLSTM layers. The fusion strategy is vector concatenation. The classification head is a Multi-Layer Perceptron (MLP).

MM-BERT, MM-RoBERTa (Mancini et al., 2024) The text module comprises a trainable text embedding model and a dropout layer on top. The audio module comprises a pre-trained audio embedding model, a BiLSTM layer, and a dropout layer. The output of the text and audio modules is concatenated and fed to the classification module, defined as a stack of dense layers.

CSA (Ours) A multimodal transformer inspired by Yu et al. (2023), whereby text and audio embeddings are concatenated along the time dimension, and a self-attention layer is employed to obtain a cross-modal text and audio embedding. This embedding is averaged over time and fed to a classification head. The main issue of this architecture is the significant difference between the lengths of the audio and text sequences. Even with downscaling, the audio embeddings tend to be significantly longer (often by a factor of $\sim$10). Consequently, audio features dominate the early stages of training, leading to underwhelming performance. To address this issue, we develop a novel transformer variant in which we reweight the attention scores of text and audio sequences for each layer. Let m be the length of the text sequence and n the length of the audio sequence, we rescale the attention scores of the text sequence by $\frac{m+n}{2m}$ and of the audio sequence by $\frac{m+n}{2n}$. This reweighting ensures that text and audio sequences have the same total weight. Figure 2 in Appendix A summarizes our Concatenation-based early fusion with Self-Attention (CSA) transformer model.

Ensemble (Ours) This architecture consists of two independent unimodal models for text and

Model	Text Encoding	Audio Encoding	Fusion
BiLSTM (Mancini et al., 2022)	GloVe + BiLSTM	(Wav2Vec2 ∨ MFCCs) + BiLSTM	Concat-Late
MM-BERT (Mancini et al., 2024)	BERT	(Wav2Vec2 ∨ HuBERT ∨ WavLM) + BiLSTM	Concat-Late
MM-RoBERTa (Mancini et al., 2024)	RoBERTa	(Wav2Vec2 ∨ HuBERT ∨ WavLM) + BiLSTM	Concat-Late
CSA (Ours)	BERT	(Wav2Vec2 ∨ HuBERT ∨ WavLM) + Transformer	Concat-Early
Ensemble (Ours)	BERT	(Wav2Vec2 ∨ HuBERT ∨ WavLM) + Transformer	Avg-Late
Mul-TA (Ours)	BERT	(Wav2Vec2 ∨ HuBERT ∨ WavLM) + Transformer	Cross

Table 1: Multimodal models available in MAMKit. For each model, we summarize its text and audio encoding modules and its fusion strategy. *Concat*: Concatenation; *Avg*: Average; *Cross*: Crossmodal Attention.

audio, respectively. A weighted average of the probability vectors of the unimodal classification heads constitutes the final prediction. The text-only model involves averaging BERT embeddings along the time dimension and feeding them to a two-layer classification head. The audio-only model follows the same architecture as the text-only model, although with a custom transformer which is trained along with the head. The main challenge is determining how to merge the outputs of the two unimodal classification heads. We compute a weighted average with weight w_e defined as follows:

$$w_e = l + (u - l) \cdot \frac{\tanh w + 1}{2} \qquad (1)$$

where w is a learnable parameter in the $[l, u]$ range. Bounding ensures that the ensemble is forced to exploit the output of both classification heads, preventing a *dead neuron* situation where the ensemble focuses on a single modality only. We set $l = 0.3$ and $u = 0.7$ for learning stability. Figure 3 in Appendix A summarizes Ensemble.

Mul-TA (Ours) We propose a variant of the MulT architecture (Tsai et al., 2019): a transformer model for carrying out multimodal tasks without the need for modality alignment. The core module of MulT is the directional pairwise crossmodal attention layer, which captures interdependencies between multimodal sequences and seamlessly adjusts information flow between modalities. In practice, the cross-modal attention layer uses one modality A as the query vector and another modality B as key and value vectors. The layer is applied for each pair of input modalities. Pairs with the same B modality are combined into a unified sequence using a self-attention layer. Lastly, each unified sequence is averaged over the time dimension and concatenated. The resulting embedding vector is fed to a classification head. While MulT was developed for three modalities, totaling six (A, B) pairs, our variant uses only two, totaling two (A, B) pairs. Additionally, we replace the self-attention unification step with an average. Figure 4 in Appendix A summarizes Mul-TA, our MulT architecture variant, tailored to text and audio modality.

3.4 Data

We now provide an overview of MAM datasets currently available in MAMKit.

UKDebates (Lippi and Torroni, 2016a) The first MAM dataset. It contains transcriptions and audio sequences of three candidates for UK Prime Ministerial elections of 2015 in a two-hour debate aired by Sky News. The candidates are David Cameron, Nick Clegg, and Ed Miliband. The dataset contains 386 sentences and corresponding audio samples. Two domain experts annotated sentences as containing or not containing a claim. The inter-annotator agreement measured via Cohen's kappa (Carletta, 1996) is 0.53 (*fair to good*).

M-Arg (Mestre et al., 2021b) A multimodal dataset built around the 2020 US Presidential elections. The dataset contains transcriptions and audio sequences of four candidates and a debate moderator concerning 18 topics. The authors design a controlled crowdsourcing data annotation process whereby each crowd worker labels sentence pairs as describing support, attack, or no relation. In total, the dataset contains 4,104 sentence pairs with corresponding aligned audio samples. A high-quality subset of the M-Arg, M-Arg$_\gamma$, containing 2,443 sentence pairs with high agreement confidence $\gamma \geq 85\%$ is commonly considered for model evaluation.

MM-USED (Mancini et al., 2022) A multimodal extension of the dataset introduced in Haddadan et al. (2019). It contains presidential candidates' debate transcripts and corresponding audio recordings aired from 1960 to 2016. In Haddadan et al. (2019), annotators labeled text sentences as containing a claim, a premise, or neither of them.

Later Mancini et al. (2022) enriched the dataset with the audio modality and aligned text sentences to audio recording snippets. This dataset consists of 26,781 labeled sentences and corresponding audio samples covering 39 debates and 26 different speakers, making it the largest MAM resource to date.

MM-USED-fallacy (Mancini et al., 2024) A multimodal extension of the dataset introduced by Goffredo et al. (2022) about argumentative fallacies. In Goffredo et al. (2022), the authors consider the dataset curated by Haddadan et al. (2019), carry out an annotation process for labeling text spans as argumentative fallacies, and introduce a taxonomy for categorizing them. Mancini et al. (2024) enrich the existing dataset with the audio modality by first converting annotations to the sentence level and then aligning them to audio recording snippets. The dataset contains 1,891 sentences labeled as argumentative fallacies belonging to six distinct categories.

3.5 Tasks

The tasks currently supported by MAMKit are derived from literature (Lippi and Torroni, 2016b; Lawrence and Reed, 2019)

Argumentative Sentence Detection Given an input sentence x, the task of argumentative sentence detection (ASD) consists of determining whether x contains an argument (*arg*) or not (*not-arg*). We extend this definition to include component detection. For instance, the task of claim detection (Levy et al., 2014; Lippi and Torroni, 2015) consists of classifying x as containing a claim (*claim*) or not (*not-claim*).

Argumentative Component Classification Given an argumentative sentence x, the task of argumentative component classification (ACC) consists of classifying x as containing one or more argumentative components according to a reference argument model. Following the *claim-premise* argument model (Walton, 2009), ACC involves identifying claims (*claim*) and premises (*premise*) in x.

Argumentative Relation Classification Given a pair of argumentative sentences x_i and x_j, the task of argumentative relation classification (ARC) consists of classifying the pair (x_i, x_j) as yielding an argumentative relation $x_i \rightarrow x_j$ of *support*, *attack*, or *neither* if no argumentative relation exists.

Argumentative Fallacy Classification Given an argumentative sentence x identified as a fallacy, the task of argumentative fallacy classification (AFC) consists of categorizing x against a given taxonomy of fallacies.

4 Experiments

We employ MAMKit to provide a robust and reproducible overview of a significant share of the work published on MAM so far. In particular, we evaluate MAMKit supported models on all available tasks and datasets. We build our evaluation as follows. Regarding model evaluation, we compute macro F1-score except on UKDebates for which we report binary F1-score (Lippi and Torroni, 2016a). We carry out a repeated five-fold cross-validation routine for UKDebates and M-Arg$_\gamma$ using the same folds defined in Mancini et al. (2022). Similarly, we perform a repeated train and test routine for MM-USED on official data splits (Haddadan et al., 2019). We set the number of repetitions to three in both cases. Lastly, we perform a leave-one-out routine for MM-USED-fallacy Mancini et al. (2024). See Appendix B for additional details.

5 Results

Table 2 reports the best classification performance for each model (See Appendix C for all results).

UKDebates We observe no notable benefits in integrating the audio modality in all models, comparable to the results reported in Mancini et al. (2022). Specifically, multimodal models show equal or lesser classification performance than their text-only modules.

M-Arg$_\gamma$ Our results significantly differ from those reported in Mancini et al. (2022). In particular, Ensemble and Mul-TA, are noticeably underperforming compared to their text-only counterparts. The only exceptions are MM-BERT and CSA with slightly higher performance. Additionally, audio-only models fail to learn the task.

MM-USED We observe a small performance gap between audio-only and text-only models, suggesting that the audio modality may be a valuable indicator in both ASD and ACC tasks. However, multimodal models achieve comparable performance to their text-only counterparts, with minor improvements only for MM-BERT (+1.7), CSA (+0.9), Ensemble (+0.2) and Mul-TA (+1.3) in ASD, CSA (+1.4), and Mul-TA (+1.8) in ACC.

Model	UKDebates (ASD)	M-Arg$_\gamma$ (ARC)	MM-USED (ASD)	MM-USED (ACC)	MM-USED-fallacy (AFC)
			Text Only		
BiLSTM (T_1)	$.552_{\pm.047}$	$.120_{\pm.006}$	$.811_{\pm.004}$	$.663_{\pm.002}$	$.525_{\pm.113}$
BERT (T_2)	$.654_{\pm.003}$	$.132_{\pm.004}$	$.824_{\pm.009}$	$.679_{\pm.004}$	$.594_{\pm.122}$
RoBERTa (T_3)	$.692_{\pm.005}$	$.172_{\pm.015}$	$.839_{\pm.010}$	$.680_{\pm.001}$	$.615_{\pm.097}$
			Audio Only		
BiLSTM (A_1)	$.393_{\pm.040}$	$.024_{\pm.012}$	$.774_{\pm.008}$	$.596_{\pm.005}$	$.657_{\pm.000}$
Transformer (A_2)	$.455_{\pm.004}$	$.000_{\pm.000}$	$.771_{\pm.019}$	$.526_{\pm.004}$	$.629_{\pm.162}$
			Text Audio		
BiLSTM ($T_1 + A_1$)	$.533_{\pm.009}$	$.084_{\pm.016}$	$.815_{\pm.006}$	$.667_{\pm.000}$	$.572_{\pm.099}$
MM-BERT ($T_2 + A_1$)	$.662_{\pm.004}$	$.160_{\pm.015}$	$.841_{\pm.005}$	$.680_{\pm.004}$	$.599_{\pm.128}$
MM-RoBERTa ($T_3 + A_1$)	$.687_{\pm.010}$	$.178_{\pm.012}$	$.837_{\pm.009}$	$.678_{\pm.003}$	$.624_{\pm.074}$
CSA ($T_2 + A_2$)	$.663_{\pm.014}$	$.160_{\pm.015}$	$.833_{\pm.011}$	$.693_{\pm.001}$	$.582_{\pm.114}$
Ensemble ($T_2 + A_2$)	$.586_{\pm.015}$	$.011_{\pm.011}$	$.826_{\pm.011}$	$.681_{\pm.002}$	$.612_{\pm.134}$
Mul-TA ($T_2 + A_2$)	$.616_{\pm.019}$	$.098_{\pm.031}$	$.837_{\pm.006}$	$.697_{\pm.003}$	$.605_{\pm.110}$

Table 2: Test classification performance on MAM datasets. For each multimodal model, we report their constituting text module (T_i) and audio module (A_j).

MM-USED-fallacy In contrast to other tasks and datasets, in MM-USED-fallacy, audio-only models are the best-performing ones. The performance of text-audio models is slightly better than that of the corresponding text-only models but below that of audio-only models. Alternative fusion strategies yielded only a moderate, non-systematic improvement.

6 Conclusion

MAM is a new, exciting and largely unexplored research domain with interesting applications. We believe that, at present, an open and collaborative standardized platform for experimentation and benchmarking has the potential to build a stronger community around it, that will be able to focus on the innovations needed to push the envelope. To this end, we developed an open-source PyTorch toolkit named MAMKit. MAMKit offers several datasets, state-of-the-art models, and processing strategies. This paper introduces the platform and discusses some initial empirical results we obtained with it.

Remarkably, the advanced audio encoding and fusion techniques we introduced do not yield the performance improvement we hoped for. This result might be ascribed to weaknesses in the architectures, and motivate further research on novel encoding and fusion methods. However, the negative result might also be attributed to the fact that, in the available datasets, annotations were first made on the transcripts, and only later extended to the audio modality. As noted by Mancini et al. (2024), such a procedure does not exploit acoustic insights, hence it should be expected that the potential of MAM architectures may not be fully leveraged, until datasets become available, that natively include auditory cues in the annotation process. This issue affects all MAM datasets in MAMKit, therefore a revision of the existing annotations would be required to effectively include auditory cues.

In conclusion, further research is needed to understand audio characteristics better and devise methods to integrate them with textual annotations. That will necessitate collaboration across fields like argumentation and signal processing. MAMKit could be a valuable resource for fostering such a collaboration. In a broader perspective, MAMKit holds potential for further development and application, including its extension to additional modalities like images and video (Birdsell and Groarke, 1996). For instance, we plan to incorporate the ImageArg dataset (Liu et al., 2022), which has been developed to address argument stance classification and image persuasiveness classification tasks. The ImageArg dataset was notably extended during ImageArg-2023 (Liu et al., 2023), the first shared task in MAM, providing additional annotated samples. This dataset has been leveraged in various studies (Sharma et al., 2023; Zong et al., 2023) proposing diverse strategies for vision-language MAM, thereby presenting an opportunity for integrating new models within MAMKit. Additionally, we plan to include in MAMKit the MMClaims dataset (Cheema et al., 2022), designed for multimodal claim detection in social media.

Furthermore, we aim to improve our understanding of multimodal discourse analysis and its practical implications through further experimentation with new datasets and by exploring transfer learning techniques to enhance model generalization across diverse domains.

7 Limitations

PyTorch Dependency. Currently, the toolkit only supports PyTorch. While PyTorch is a widely used deep learning framework, this limitation may pose challenges for researchers who prefer or require other frameworks, such as TensorFlow, as well as the integration of previous work built on these frameworks.

Incomplete Dataset and Model Integration. Not all existing datasets and models for MAM research are included. For instance, the VivesDebate-Speech dataset (Ruiz-Dolz and Iranzo-Sánchez, 2023), the ImageArg dataset (Liu et al., 2022), the MMClaims dataset (Cheema et al., 2022) and models like M-ArgNet (Mestre et al., 2021b) are currently not implemented. We plan to integrate these and other resources in the future, and we encourage MAM researchers to include their resources on our platform.

Scope Limitation. At present, the toolkit focuses solely on text and audio modalities. We recognize the importance of expanding to other modalities, such as visual AM. Resources for these additional modalities will be integrated in future work.

References

2020. Large-scale self-supervised pre-training across tasks, languages, and modalities.

2022. Torchmultimodal (beta release).

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. In *Advances in Neural Information Processing Systems*, volume 33, pages 12449–12460. Curran Associates, Inc.

M. Sahbi Benlamine, Maher Chaouachi, Serena Villata, Elena Cabrio, Claude Frasson, and Fabien Gandon. 2015. Emotions in argumentation: an empirical evaluation. In *IJCAI*, pages 156–163. AAAI Press.

David S. Birdsell and Leo Groarke. 1996. Toward a theory of visual argument. *Argumentation and advocacy*, 33(1):1.

Said Yacine Boulahia, Abdenour Amamra, Mohamed Ridha Madi, and Said Daikh. 2021. Early, intermediate and late fusion strategies for robust deep learning-based multimodal action recognition. *Machine Vision and Applications*, 32(6):121.

Hervé Bredin. 2017. pyannote.metrics: A Toolkit for Reproducible Evaluation, Diagnostic, and Error Analysis of Speaker Diarization Systems. In *Proc. Interspeech 2017*, pages 3587–3591.

Hervé Bredin. 2023. pyannote.audio 2.1 speaker diarization pipeline: principle, benchmark, and recipe. In *Proc. INTERSPEECH 2023*, pages 1983–1987.

Mikhail Burtsev, Alexander Seliverstov, Rafael Airapetyan, Mikhail Arkhipov, Dilyara Baymurzina, Nickolay Bushkov, Olga Gureenkova, Taras Khakhulin, Yuri Kuratov, Denis Kuznetsov, Alexey Litinsky, Varvara Logacheva, Alexey Lymar, Valentin Malykh, Maxim Petrov, Vadim Polulyakh, Leonid Pugachev, Alexey Sorokin, Maria Vikhreva, and Marat Zaynutdinov. 2018. DeepPavlov: Open-source library for dialogue systems. In *Proceedings of ACL 2018, System Demonstrations*, pages 122–127, Melbourne, Australia. Association for Computational Linguistics.

Jean Carletta. 1996. Assessing agreement on classification tasks: The kappa statistic. *Comput. Linguistics*, 22(2):249–254.

Gullal Singh Cheema, Sherzod Hakimov, Abdul Sittar, Eric Müller-Budack, Christian Otto, and Ralph Ewerth. 2022. MM-claims: A dataset for multimodal claim detection in social media. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 962–979, Seattle, United States. Association for Computational Linguistics.

Sanyuan Chen, Chengyi Wang, Zhengyang Chen, Yu Wu, Shujie Liu, Zhuo Chen, Jinyu Li, Naoyuki Kanda, Takuya Yoshioka, Xiong Xiao, Jian Wu, Long Zhou, Shuo Ren, Yanmin Qian, Yao Qian, Jian Wu, Michael Zeng, Xiangzhan Yu, and Furu Wei. 2022. Wavlm: Large-scale self-supervised pre-training for full stack speech processing. *IEEE Journal of Selected Topics in Signal Processing*, 16(6):1505–1518.

Jean-benoit Delbrouck, Khaled Saab, Maya Varma, Sabri Eyuboglu, Pierre Chambon, Jared Dunnmon, Juan Zambrano, Akshay Chaudhari, and Curtis Langlotz. 2022. ViLMedic: a framework for research at the intersection of vision and language in medical AI. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 23–34, Dublin, Ireland. Association for Computational Linguistics.

Victor Dibia. 2020. NeuralQA: A usable library for question answering (contextual query expansion + BERT) on large datasets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 15–22, Online. Association for Computational Linguistics.

William Falcon and The PyTorch Lightning team. 2019. Pytorchlightning.

Pierpaolo Goffredo, Shohreh Haddadan, Vorakit Vorakitphan, Elena Cabrio, and Serena Villata. 2022. Fallacious argument classification in political debates. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 4143–4149. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Leo Groarke and Gabrijela Kišiček. 2018. Sound arguments: An introduction to auditory argument. In *Argumentation and inference: Proceedings of 2nd European Conference on Argumentation*, pages 177–198. London: Collage Publications.

Shohreh Haddadan, Elena Cabrio, and Serena Villata. 2019. Yes, we can! mining arguments in 50 years of US presidential campaign debates. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4684–4690, Florence, Italy. Association for Computational Linguistics.

Chadi Helwe, Chloé Clavel, and Fabian Suchanek. 2022. LogiTorch: A PyTorch-based library for logical reasoning on natural language. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 250–257, Abu Dhabi, UAE. Association for Computational Linguistics.

Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. 2021. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 29:3451–3460.

Andreea Iana, Goran Glavaš, and Heiko Paulheim. 2023. NewsRecLib: A PyTorch-lightning library for neural news recommendation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 296–310, Singapore. Association for Computational Linguistics.

Petar Ivanov, Ivan Koychev, Momchil Hardalov, and Preslav Nakov. 2023. Detecting check-worthy claims in political debates, speeches, and interviews using audio data. *CoRR*, abs/2306.05535.

Gabrijela Kišiček. 2014. The role of prosodic features in the analysis of multimodal argumentation. In *International Society for the Study of Argumentation (ISSA), 8th international conference on argumentation*. Rozenberg Quarterly, The Magazine.

John Lawrence and Chris Reed. 2019. Argument mining: A survey. *Computational Linguistics*, 45(4):765–818.

Ran Levy, Yonatan Bilu, Daniel Hershcovich, Ehud Aharoni, and Noam Slonim. 2014. Context dependent claim detection. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*, pages 1489–1500. ACL.

Dongxu Li, Junnan Li, Hung Le, Guangsen Wang, Silvio Savarese, and Steven C.H. Hoi. 2023. LAVIS: A one-stop library for language-vision intelligence. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 31–41, Toronto, Canada. Association for Computational Linguistics.

Yehao Li, Yingwei Pan, Jingwen Chen, Ting Yao, and Tao Mei. 2021. X-modaler: A versatile and high-performance codebase for cross-modal analytics. In *Proceedings of the 29th ACM International Conference on Multimedia*, MM '21, page 3799–3802, New York, NY, USA. Association for Computing Machinery.

Marco Lippi and Paolo Torroni. 2015. Context-independent claim detection for argument mining. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 185–191. AAAI Press.

Marco Lippi and Paolo Torroni. 2016a. Argument mining from speech: Detecting claims in political debates. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 2979–2985. AAAI Press.

Marco Lippi and Paolo Torroni. 2016b. Argumentation mining: State of the art and emerging trends. *ACM Trans. Internet Technol.*, 16(2).

Zhexiong Liu, Mohamed Elaraby, Yang Zhong, and Diane Litman. 2023. Overview of ImageArg-2023: The first shared task in multimodal argument mining. In *Proceedings of the 10th Workshop on Argument Mining*, pages 120–132, Singapore. Association for Computational Linguistics.

Zhexiong Liu, Meiqi Guo, Yue Dai, and Diane Litman. 2022. ImageArg: A multi-modal tweet dataset for image persuasiveness mining. In *Proceedings of the 9th Workshop on Argument Mining*, pages 1–18, Online and in Gyeongju, Republic of Korea. International Conference on Computational Linguistics.

Vidula Magdum, Omkar Jayant Dhekane, Sharayu Sandeep Hiwarkhedkar, Saloni Sunil Mittal, and Raviraj Joshi. 2023. mahaNLP: A Marathi natural language processing library. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 34–40, Bali, Indonesia. Association for Computational Linguistics.

Eleonora Mancini, Federico Ruggeri, Andrea Galassi, and Paolo Torroni. 2022. Multimodal argument mining: A case study in political debates. In *Proceedings* *of the 9th Workshop on Argument Mining*, pages 158–170, Online and in Gyeongju, Republic of Korea. International Conference on Computational Linguistics.

Eleonora Mancini, Federico Ruggeri, and Paolo Torroni. 2024. Multimodal fallacy classification in political debates. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 170–178, St. Julian's, Malta. Association for Computational Linguistics.

Rafael Mestre, Stuart E. Middleton, Matt Ryan, Masood Gheasi, Timothy Norman, and Jiatong Zhu. 2023. Augmenting pre-trained language models with audio feature embedding for argumentation mining in political debates. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 274–288, Dubrovnik, Croatia. Association for Computational Linguistics.

Rafael Mestre, Razvan Milicin, Stuart Middleton, Matt Ryan, Jiatong Zhu, and Timothy J. Norman. 2021a. M-arg: Multimodal argument mining dataset for political debates with audio and transcripts. In *ArgMining@EMNLP*, pages 78–88. Association for Computational Linguistics.

Rafael Mestre, Razvan Milicin, Stuart E. Middleton, Matt Ryan, Jiatong Zhu, and Timothy J. Norman. 2021b. M-arg: Multimodal argument mining dataset for political debates with audio and transcripts. In *Proceedings of the 8th Workshop on Argument Mining*, pages 78–88, Punta Cana, Dominican Republic. Association for Computational Linguistics.

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. TextAttack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126, Online. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Ramon Ruiz-Dolz and Javier Iranzo-Sánchez. 2023. VivesDebate-speech: A corpus of spoken argumentation to leverage audio features for argument mining. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2071–2077, Singapore. Association for Computational Linguistics.

Arushi Sharma, Abhibha Gupta, and Maneesh Bilalpur. 2023. Argumentative stance prediction: An exploratory study on multimodality and few-shot learning. In *Proceedings of the 10th Workshop on Argument Mining*, pages 167–174, Singapore. Association for Computational Linguistics.

Jiatong Shi, Shuai Guo, Tao Qian, Tomoki Hayashi, Yuning Wu, Fangzheng Xu, Xuankai Chang, Huazhe Li, Peter Wu, Shinji Watanabe, and Qin Jin. 2022. Muskits: an End-to-end Music Processing Toolkit for Singing Voice Synthesis. In *Proc. Interspeech 2022*, pages 4277–4281.

Amanpreet Singh, Vedanuj Goswami, Vivek Natarajan, Yu Jiang, Xinlei Chen, Meet Shah, Marcus Rohrbach, Dhruv Batra, and Devi Parikh. 2020. Mmf: A multimodal framework for vision and language research. `https://github.com/facebookresearch/mmf`.

Felix Stollenwerk. 2023. nerblackbox: A high-level library for named entity recognition in python. In *Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023)*, pages 174–178, Singapore. Association for Computational Linguistics.

Tianyi Tang, Junyi Li, Zhipeng Chen, Yiwen Hu, Zhuohao Yu, Wenxun Dai, Wayne Xin Zhao, Jian-yun Nie, and Ji-rong Wen. 2022. TextBox 2.0: A text generation library with pre-trained language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 435–444, Abu Dhabi, UAE. Association for Computational Linguistics.

Ermal Toto, ML Tlachac, and Elke A. Rundensteiner. 2021. Audibert: A deep transfer learning multimodal classification framework for depression screening. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, page 4145–4154, New York, NY, USA. Association for Computing Machinery.

Yao-Hung Hubert Tsai, Shaojie Bai, Paul Pu Liang, J. Zico Kolter, Louis-Philippe Morency, and Ruslan Salakhutdinov. 2019. Multimodal transformer for unaligned multimodal language sequences. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6558–6569, Florence, Italy. Association for Computational Linguistics.

Douglas Walton. 2009. *Argumentation Theory: A Very Short Introduction*, pages 1–22. Springer US, Boston, MA.

Xin Xie, Zhoubo Li, Xiaohan Wang, ZeKun Xi, and Ningyu Zhang. 2023. LambdaKG: A library for pre-trained language model-based knowledge graph embeddings. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 25–33, Bali, Indonesia. Association for Computational Linguistics.

Tianshu Yu, Haoyu Gao, Ting-En Lin, Min Yang, Yuchuan Wu, Wentao Ma, Chao Wang, Fei Huang, and Yongbin Li. 2023. Speech-text pre-training for spoken dialog understanding with explicit cross-modal alignment. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7900–7913, Toronto, Canada. Association for Computational Linguistics.

Guoyang Zeng, Fanchao Qi, Qianrui Zhou, Tingji Zhang, Zixian Ma, Bairu Hou, Yuan Zang, Zhiyuan Liu, and Maosong Sun. 2021. OpenAttack: An open-source textual adversarial attack toolkit. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 363–371, Online. Association for Computational Linguistics.

Qing Zong, Zhaowei Wang, Baixuan Xu, Tianshi Zheng, Haochen Shi, Weiqi Wang, Yangqiu Song, Ginny Wong, and Simon See. 2023. TILFA: A unified framework for text, image, and layout fusion in argument mining. In *Proceedings of the 10th Workshop on Argument Mining*, pages 139–147, Singapore. Association for Computational Linguistics.

Appendix

A Model Architectures

We provide a comprehensive visual representation of the novel model architectures presented in this work. Figures 2, 3, and 4 show the CSA, Ensemble, and Mul-TA models, respectively.

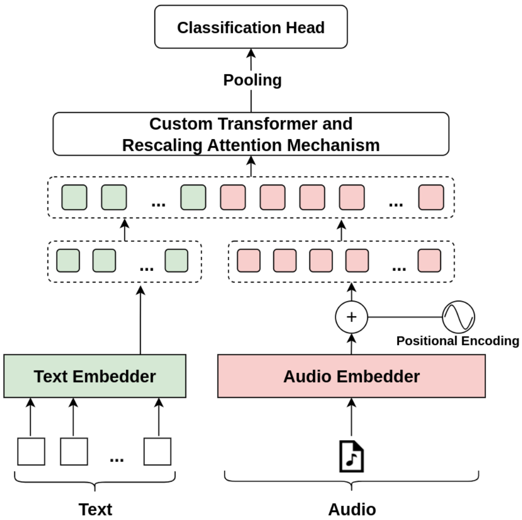

Figure 2: The CSA model architecture.

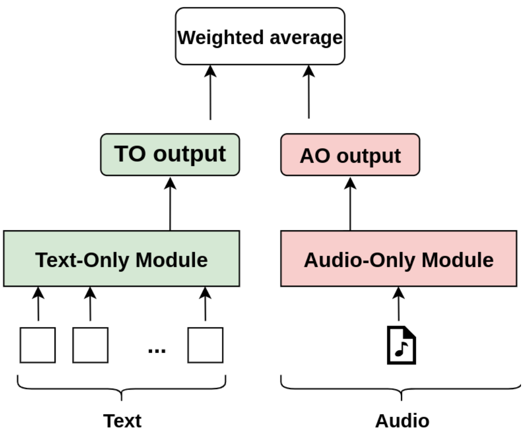

Figure 3: The Ensemble model architecture.

B Experimental Setup Details

Model Hyper-parameters Table 3 reports the main hyper-parameters used in our experiments. All model configurations can be inspected in the `mamkit.configs` package.

Training Models are trained with cross-entropy loss as standard practice for classification tasks. We additionally apply class weighting to address class imbalance in all datasets except for MM-USED ACC, where weighting is not needed. We monitor validation loss during training and load the best model checkpoint based on this metric for evaluation on validation and test splits.

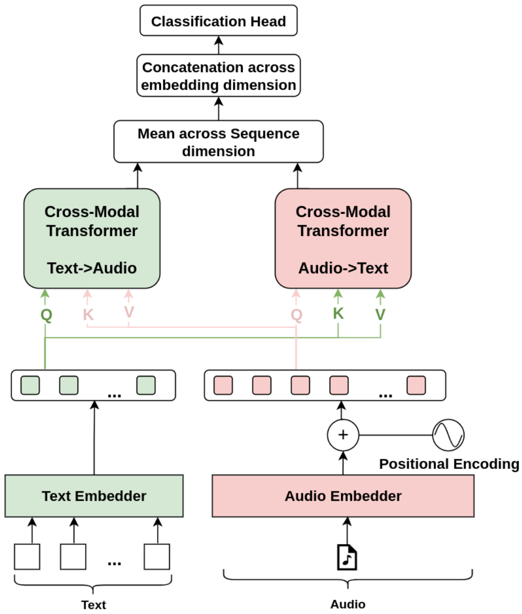

Figure 4: The Mul-TA model architecture.

General	
`optimizer`	AdamW
`batch_size`	4
`gradient accumulation steps`	3
`effective batch_size`	12
`max_epochs`	20
`early_stopping patience`	5
`early_stopping monitor`	*val_loss*
`cross-validation seeds`	42, 2024, 666
`leave-one-out seeds`	42
`train and test seeds`	42, 2024, 666

Table 3: General model hyper-parameters in our experiments.

Hardware We employ an NVIDIA 2080Ti GPU with 12 GB VRAM and an NVIDIA 3060Ti GPU with 8 GB VRAM to run our experiments. All experiments regarding a dataset are run on the same device for reproducibility and fair comparison. Furthermore, individual experiments were run on a single device.

C Additional Results

Table 4 reports results for all model combinations evaluated.

Model	**UKDebates (ASD)**	**M-Arg$_\gamma$ (ARC)**	**MM-USED (ASD)**	**MM-USED (ACC)**	**MMUSED-fallacy (AFC)**
			Text Only		
BiLSTM	$.552_{\pm.047}$	$.120_{\pm.006}$	$.811_{\pm.004}$	$.663_{\pm.002}$	$.525_{\pm.113}$
BERT	$.654_{\pm.003}$	$.132_{\pm.004}$	$.824_{\pm.009}$	$.679_{\pm.004}$	$.594_{\pm.122}$
RoBERTa	$.692_{\pm.005}$	$.172_{\pm.015}$	$.839_{\pm.010}$	$.680_{\pm.001}$	$.615_{\pm.097}$
			Audio Only		
BiLSTM w/ MFCCs	$.302_{\pm.047}$	$.003_{\pm.005}$	$.722_{\pm.027}$	$.527_{\pm.004}$	$\mathbf{.657}_{\pm\mathbf{.000}}$
BiLSTM w/ Wav2Vec2	$.376_{\pm.023}$	$.000_{\pm.000}$	$\mathbf{.774}_{\pm\mathbf{.008}}$	$\mathbf{.596}_{\pm\mathbf{.005}}$	$.655_{\pm.117}$
BiLSTM w/ HuBERT	$.364_{\pm.012}$	$\mathbf{.024}_{\pm\mathbf{.012}}$	$.745_{\pm.009}$	$.566_{\pm.007}$	$.638_{\pm.000}$
BiLSTM w/ WavLM	$\mathbf{.393}_{\pm\mathbf{.040}}$	$.010_{\pm.010}$	$.772_{\pm.015}$	$.583_{\pm.002}$	$.652_{\pm.000}$
Transformer w/ Wav2Vec2	$.440_{\pm.030}$	$.000_{\pm.000}$	$\mathbf{.771}_{\pm\mathbf{.019}}$	$.514_{\pm.000}$	$.567_{\pm.225}$
Transformer w/ HuBERT	$.425_{\pm.033}$	$.000_{\pm.000}$	$.765_{\pm.016}$	$.524_{\pm.004}$	$\mathbf{.629}_{\pm\mathbf{.162}}$
Transformer w/ WavLM	$\mathbf{.455}_{\pm\mathbf{.004}}$	$.000_{\pm.000}$	$.768_{\pm.005}$	$\mathbf{.526}_{\pm\mathbf{.004}}$	$.594_{\pm.217}$
			Text Audio		
BiLSTM w/ MFCCs	$.528_{\pm.039}$	$.065_{\pm.014}$	$.807_{\pm.010}$	$.662_{\pm.006}$	$\mathbf{.572}_{\pm\mathbf{.099}}$
BiLSTM w/ Wav2Vec2	$\mathbf{.533}_{\pm\mathbf{.009}}$	$.079_{\pm.014}$	$.808_{\pm.012}$	$.665_{\pm.004}$	$.505_{\pm.168}$
BiLSTM w/ HuBERT	$.409_{\pm.017}$	$.055_{\pm.020}$	$.807_{\pm.013}$	$.653_{\pm.003}$	$.456_{\pm.131}$
BiLSTM w/ WavLM	$.501_{\pm.022}$	$\mathbf{.084}_{\pm\mathbf{.016}}$	$\mathbf{.815}_{\pm\mathbf{.006}}$	$\mathbf{.667}_{\pm\mathbf{.000}}$	$.526_{\pm.174}$
MM-BERT w/ Wav2Vec2	$\mathbf{662}_{\pm\mathbf{.004}}$	$.153_{\pm.017}$	$\mathbf{841}_{\pm\mathbf{.005}}$	$.677_{\pm.003}$	$.561_{\pm.114}$
MM-BERT w/ HuBERT	$.626_{\pm.003}$	$\mathbf{.160}_{\pm\mathbf{.015}}$	$.840_{\pm.006}$	$.677_{\pm.004}$	$\mathbf{.599}_{\pm\mathbf{.128}}$
MM-BERT w/ WavLM	$.654_{\pm.019}$	$.152_{\pm.008}$	$.836_{\pm.005}$	$\mathbf{.680}_{\pm\mathbf{.004}}$	$.580_{\pm.103}$
MM-RoBERTa w/ Wav2Vec2	$.674_{\pm.009}$	$\mathbf{.178}_{\pm\mathbf{.012}}$	$.833_{\pm.006}$	$\mathbf{.678}_{\pm\mathbf{.003}}$	$.608_{\pm.126}$
MM-RoBERTa w/ HuBERT	$.624_{\pm.015}$	$.147_{\pm.004}$	$.837_{\pm.003}$	$.677_{\pm.008}$	$.576_{\pm.097}$
MM-RoBERTa w/ WavLM	$\mathbf{.687}_{\pm\mathbf{.010}}$	$.165_{\pm.018}$	$\mathbf{.837}_{\pm\mathbf{.009}}$	$\mathbf{678}_{\pm\mathbf{.003}}$	$\mathbf{.624}_{\pm\mathbf{.074}}$
CSA w/ Wav2Vec2	$\mathbf{.663}_{\pm\mathbf{.014}}$	$.137_{\pm.027}$	$.822_{\pm.002}$	$\mathbf{.693}_{\pm\mathbf{.001}}$	$.555_{\pm.118}$
CSA w/ HuBERT	$.632_{\pm.018}$	$\mathbf{.160}_{\pm\mathbf{.015}}$	$.813_{\pm.004}$	$\mathbf{.693}_{\pm\mathbf{.001}}$	$\mathbf{.582}_{\pm\mathbf{.114}}$
CSA w/ WavLM	$.655_{\pm.029}$	$.155_{\pm.030}$	$\mathbf{.833}_{\pm\mathbf{.011}}$	$.697_{\pm.001}$	$.535_{\pm.102}$
Ensemble w/ Wav2Vec2	$\mathbf{.586}_{\pm\mathbf{.015}}$	$\mathbf{.011}_{\pm\mathbf{.011}}$	$.825_{\pm.004}$	$\mathbf{.681}_{\pm\mathbf{.002}}$	$\mathbf{.612}_{\pm\mathbf{.134}}$
Ensemble w/ HuBERT	$.531_{\pm.039}$	$.010_{\pm.004}$	$.822_{\pm.007}$	$.681_{\pm.003}$	$.611_{\pm.107}$
Ensemble w/ WavLM	$.576_{\pm.006}$	$.002_{\pm.003}$	$\mathbf{.826}_{\pm\mathbf{.011}}$	$.680_{\pm.003}$	$.605_{\pm.136}$
Mul-TA w/ Wav2Vec2	$.592_{\pm.034}$	$\mathbf{.098}_{\pm\mathbf{.031}}$	$.826_{\pm.011}$	$.695_{\pm.001}$	$\mathbf{.605}_{\pm\mathbf{.110}}$
Mul-TA w/ HuBERT	$\mathbf{.616}_{\pm\mathbf{.019}}$	$.079_{\pm.053}$	$.829_{\pm.011}$	$\mathbf{.697}_{\pm\mathbf{.003}}$	$.594_{\pm.091}$
Mul-TA w/ WavLM	$.602_{\pm.017}$	$.063_{\pm.015}$	$\mathbf{.837}_{\pm\mathbf{.006}}$	$.690_{\pm.003}$	$.605_{\pm.082}$

Table 4: Test classification performance on MAM datasets. In bold, the best-performing model for each configuration.