A Cross-Domain Evaluation of Multimodal Argument Relation Identification

Oscar Morris
ID: 2497790
AC40001 Honours Project
BSc (Hons) Computing Science
University of Dundee
Supervisor: Dr. R. Ruiz-Dolz
April 2025

1 Introduction

2 Background

2.1 Argumentation Theory

Argument and debate has been studied since the time of the ancient Greek philosophers and rhetoricians where argument theorists have sought to formalise discourse and discover some standard of proof for determining the 'correctness' of an argument. Over time, theories of arguments and discussions have evolved, notably when Hamblin [1] refashioned an argumentative discourse as a game, where one party makes moves offering premises that may be acceptable to the another party in the discourse who doubts the conclusion of the argument.

In order to describe various dialogue, argument and illocutionary structures different models (annotation schemes) can be used, some annotation schemes focus on types of the text itself (such as speech act theory [2]) or on the types of relations between components (such as Rhetorical Structure Theory [3]). Inference Anchoring Theory (IAT) [4] is an annotation scheme constructed to benefit from insights across both types, whilst focusing specifically on argumentative discourse. This makes IAT a very useful tool to analyse arguments and their relations.

In IAT, the discourse is first segmented into Argumentative Discourse Units (ADUs). An ADU is any span of the discourse which has both propositional content and discrete argumentative function [5]. An IAT argument graph is typically composed of two main parts: the left-hand side and the right-hand side. The right-hand side is concerned with locutions and transitions between them. A locution is simply the text of the ADU as uttered, without reconstructing ellipses or resolving pronouns. Locutions also include the speaker and may even include a timestamp. Transitions connect locutions capturing a functional relationship between predecessor and successor locutions (i.e. a response or reply).

The left-hand side of an argument graph is more concerned with the content of the ADU, rather than directly reflecting what was uttered. This consists of the propositions made, and the relations between those propositions. To create a proposition from an ADU, the content is reconstructed to be a coherent, lone-standing sentence. This means that any missing or implicit material has to be reconstructed, including anaphoric references (e.g. pronouns).

IAT defines three different types of propositional relation: *inference*, *conflict* and *rephrase*. An inference relation (also termed RA) holds between two propositions when one (the premise) is used to provide a reason to accept the other (the conclusion). This may include annotation of the kind of support e.g. Modus Ponens or Argument from Expert Opinion. These subtypes of relation are often called *argument schemes* [6], [7]. There are also several different inference structures:

- **Serial arguments** occur when one proposition supports another, which in turn supports a third.
- **Convergent arguments** occur when multiple premises act independently to support the same conclusion.
- **Linked arguments** occur when multiple premises work together to support a conclusion.

- **Divergent arguments** occur when a single premise is used to support multiple conclusions.

A conflict relation (also termed CA) holds between two propositions when one is used to provide an incompatible alternative to another and can also be of a given kind (e.g. Conflict from Bias, Conflict from Propositional Negation). The following conflict structures are identified by IAT:

- **Rebutting conflict** occurs if one proposition is directly targeting another by indicating that the latter is not acceptable.
- **Undermining conflict** occurs if a conflict is targeting the premise of an argument, then it is undermining its conclusion.
- **Undercutting conflict** occurs if the conflict is targeting the inference relation between two propositions.

A rephrase relation (also termed MA) holds when one proposition rephrases, restates or reformulates another but with dfferent propositional content (i.e. one proposition cannot simply repeat the other). There are many different kinds of rephrase, such as Specialisation, Generalisation, Instantiation etc. Generally, question answering will often involve a rephrase because the propositional content of the question is typically instantiated, resolved or refined by its answer. In contrast to inference, conflict and rephrase structures only have a single incoming an one outgoing edge.

The left and right-hand sides are connected by *illocutionary connections*. These illocutionary connections are based on illocutionary force as introduced by speech act theory [2]. The speech act $F(p)$ is the act which relates the locution and and the propositional content p through the illocutionary force F e.g. asserting p, requesting p, promising p etc. There are many diferent types of illocutionary connection, including: assertions, questions, challenges, concessions and (dis-)affirmations [8].

There have been several ways to store argumentative data created, for example Argument Markup Language (AML) [9], an XML-based language used to describe arguments in the Araucaria software. More recently, the Argument Interchange Format (AIF) [10] has been created to standardise the storage of IAT graphs.

AIF treats all relevant parts of the argument as nodes within a graph. These nodes can be put into two categories: *information nodes* (I-nodes) and *scheme nodes* (S-nodes). I-nodes represent the claims made in the discourse whereas S-nodes indicate the application of an argument scheme. Initially I-nodes only included the propositions made [10], but when Reed *et al.* [11] extended AIF to cater to dialogues, they added L-nodes as a subclass of I-nodes to represent locutions. For the purposes of this research, I-nodes and L-nodes are considered separate classes where I-nodes contain propositions and L-nodes contain locutions.

Since AIF data can be easily shared, it became the basis for a Worldwide Argument Web (WWAW) [12]. Since then, many corpora have been annotated using IAT and published on the AIFdb[1] [13] providing a very useful resource for argumentation research of many kinds.

2.2 Natural Language Processing

In recent years there have been several major advances in the field of natural language processing (NLP), most notably the introduction of the transformer architecture [14]. The transformer architecture, based on self-attention, allows the model to determine much longer range dependencies than previous approaches.

Even before Vaswani *et al.* introduced the transformer architecture supervised and semi-supervised pre-training approaches were already being explored, and proven to be a very useful tool for improving the performance of language models [15], [16]. When the transformer was introduced these pre-training techinques were adapted for use in transformers creating models which are able to be fine-tuned with relatively minimal effort and compute to allow high performance on a wide variety of tasks [17], [18]. The pre-training approaches introduced by BERT and RoBERTa use a combination of masked language modelling (where the model is trained to predict the token hidden under a `[mask]` token) and next sentence prediction. The models are then trained using this approach on a large amount of data (the data used to

[1] https://www.aifdb.org/

pre-train RoBERTa totals over 160GB of uncompressed text).

A similar progression can be seen in the development of audio models. Pre-training was notably introduced into speech recognition with wav2vec [19], where the model is trained to predict future samples from a given signal. The wav2vec model has two main stages, first raw audio samples are fed into a convolutional network which performs a similar role to the tokenisation seen in text-based language models by using a sliding window approach to downsample the audio data. These encodings are then fed into a second convolutional network to create a final encoding for the sequence.

Transformer models were introduced into the architecture of audio models with wav2vec2 [20] and HuBERT [21], where the second convolutional model is replaced with a transformer in order to better learn dependencies across the entire sequence. These models are then pre-trained on significant amounts of audio data (960 hours in the case of wav2vec2) in order to then be fine-tuned on a downstream task.

Transformer models have recently become much more well-known due to the introduction of Large Language Models (LLMs) such as GPT-4 [22] and LLaMA [23]. LLMs have proven very useful across NLP due to their ability to achieve high performance on many tasks without the need for fine-tuning, this can, however, include few-shot techniques to allow them to 'learn' at inference time [24], [25].

Combining modalities (such as text and audio) has also proven to be a useful tool across several tasks, including medical imaging [26], [27] and natural language processing [28], [29], including argument mining [30], [31]. Generally fusion techniques can be split into two categories: early and late. Early fusion techniques combine representations of each modality before being used as input to an encoder, with the primary benefit that only a single encoder is used. Late fusion techniques use a separate encoder for each modality, and the encodings are then fused to provide a crossmodal representation of the input.

In early fusion the input representations are transformed into a common information space, often using vectorisation techniques dependent on the modality. Late fusion techiques allow for the encodings of each modality to be combined in several different ways, often either simple operations (such as concatenation or an element-wise product) but a cross-modal attention module can also be used to combine the modalities [32], [33]. The fusion techniques used in this project are explained in detail in Section 4.

2.3 Argument Mining

Argument Mining (AM) is the automatic identification and extraction of inference and reasoning in natural language [34]. Various NLP techniques have been beneficial to AM, including Support Vector Machines, and more recently neural networks, in particular the transformer architecture [35]. Before discussing the automation of AM, it is useful to understand how argument analysis is conducted manually. Manual argument analysis considers the following steps:

- **Text Segmentation** involves the splitting of the original text/discourse into the pieces that will form the resulting argument structure. These pieces are often termed Elementary Discourse Units (EDUs).
- **Argument / Non-Argument Classification** is the task of determining which of the segments found in the text segmentation step are relevant to the argument. For most manual analysis, this step is performed in conjunction with text segmentation i.e. the analyst doesn't segment parts of the text which are not relevant to the argument.
- **Simple Structure** is the identification of relations between the arguments (e.g. inference, conflict and rephrase) and their structures (e.g. convergent, serial etc.).
- **Refined Structure** refers to the identification of argumentation schemes (e.g. Argument from Expert Opinion, Conflict From Bias etc.).

When the arument analysis process is automated, the stages are very similar to those in the manual process. Lawrence and Reed [34] define the steps as follows, increasing in computational complexity:

- **Identifying Argument Components** combines the stages of text segmentation and argument / non-

argument classification in the manual process.

- **Identifying Clausal Properties** involves the identification of both intrinsic clausal properties (e.g. is X evidence?, is X reported speech?) of the ADU and the contextual properties (e.g. is X a premise?, is X a conclusion?).
- **Identifying Relational Properties** relates to the identification of *general relations* between ADUs (e.g. is X a premise for Y?, is X in conflict with Y?) and the identification of argument schemes.

Generally these stages of AM are not directly used in the literature, but instead a set of AM sub-tasks which map onto each of these stages, the tasks defined as follows were defined by :

- **Argumentative Sentence Detection (ASD)** is the task of classifying a sequence as cntaining an argument, or not. ASD can be extended to include the task of claim detection, where a sequence is classified as containing a claim or not containing a claim.
- **Argumentative Relation Identification (ARI)** is the task of identifying the relation between a pair of sentences where given a pair (x_i, x_j) the task is to identify the argumentative relation $x_i \rightarrow x_j$ across some relation model.

There are varying relation models for ARI, the most commonly used is simply classifying the pair as one of support (a combination of inference and rephrase), attack (conflict) or unrelated. For the purposes of this project this is termed 3-class ARI. ARI can also be conducted using all relations described in IAT (inference, rephrase and conflict), as well as unrelated nodes. For the purpose of this project this is termed 4-class ARI. Some literature makes a distinction between Argument Relation Identification and Argument Relation Classification, where the latter does not involve unrelated pairs (i.e. given that the pair (x_i, x_j) is related, what is the type of relation?), however this distinction is my no means universal among AM literature [36].

Much of the AM literature only evaluates their systems in the same domain (dataset) as it was trained on [34], [37]–[41]. Recently, however, more research has been conducted into how these models perform across different domains [35], [42], [43], this generally involves training the model on one domain and then evaluating its performance across several others. A good example of this is the ARIES benchmark [36], which provides results for various different approaches to the ARI task across popular ARI datasets. Another notable contribution is Ruiz-Dolz *et al.* (2021) [35] which compares the cross-domain performance of the most popular pre-trained transformer models (e.g. BERT [17] and RoBERTa [18]) showing that the RoBERTa models tend to perform better both in-domain and cross-domain. Research has also been conducted into how these models perform cross-lingually creating a baseline for multilingual argument mining [44].

Ruiz-Dolz *et al.* (2025) [45] proposes techniques to answer the question: How do we sample unrelated arguments? If all possible examples of unrelated samples are used it constitutes an overwhelming proportion of the dataset (98-100%) which would be detrimental to model performance in the real world. To achieve this, they propose the following methods:

- **Undersampling** creates a more balanced class distribution by randomly choosing unrelated propositions from the set of all possible combinations.
- **Long Context Sampling** where unrelated propositions are chosen such that they are 'far apart' in the discourse. Ruiz-Dolz *et al.* define this as being from different argument maps.
- **Short Context Sampling** where unrelated propositons are chosen such that they are 'close together' in the discourse. Ruiz-Dolz *et al.* define this as being from the same argumet map.
- **Semantic Similarity Sampling** where unrelated propositions are chosen such that they are semantically similar.

They show that Short Context Sampling is the most challenging method when looking in-domain, however, the model is better able to generalise across different domains than the other methods and is a more realistic task.

Argument Mining techniques have also been extended to multiple modalities, both using Vision-Language systems [46]–[48] and perhaps the more obvious Audio-Language systems [31], [49], [50]. Making use of acoustic features has been proven to improve performance across both ASD and ARI tasks [30], [50] but there has not been any research into the applicability of Audio-

Language systems in cross-domain contexts.

Mancini *et al.* [31] created a comprehensive toolkit for argument mining research. They include both datasets and models that can be used for the creation and evaluation of audio-language argument mining systems, across different tasks, including ASD, ACC and ARI. Therefore, MAMKit provides a very useful benchmark for the development of audio-language AM techniques.

3 Datasets

All datasets used in this project are available as corpora on AIFdb[2]. Using consistently annotated Argument Interchange Format (AIF) data allows many different datasets to be used and tested. The AIF Format [10] allows the annotation of argument data across all AM tasks, providing a platform for many different kinds of research.

Throughout the project two primary corpora have been considered: QT30 [51], a corpus consisting of 30 AIF annotated Question Time episodes, and a corpus of 9 AIF annotated Moral Maze episodes available on AIFdb.

3.1 Preprocessing

3.1.1 Argument Data

In order to use AIF data efficiently for ARI, it is useful to perform some preprocessing. This process produces a graph, where each node contains a locution, its related proposition, and the proposition's AIF identifier. This identifier corresponds to the audio data, allowing it to be easily loaded when required. Each edge in this graph corresponds to a relation between the propositions, one of RA (inference), MA (rephrase) or CA (conflict).

Figure 1 shows an example sub-graph from the larger argument graph. Each node is truncated for brevity and only shows the node's ID, and the proposition. This sub-graph is taken from the Moral Maze episode on the 75th Anniversary of D-Day. The major downside of processing the data in this way is simply that much of the nuance encoded within AIF is lost. This is primarily the

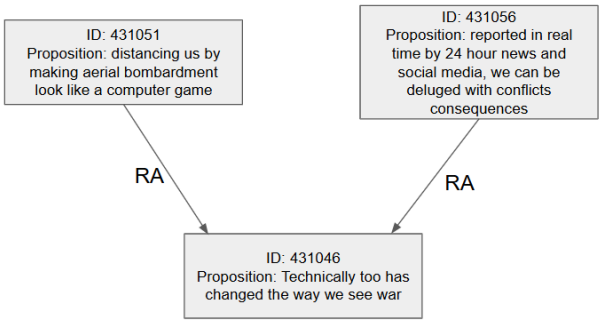

Figure 1: Example sub-graph.

inference and conflict structures (e.g. linked arguments, undercutting conflict etc.) but also the transitions and illocutionary connections. In the context of this project this is not an issue but is worth remembering when examining the data.

An example of the JSON structure used to store the argument data is shown in Listing 1. It is also worth understanding the link between the locution and the proposition, of which those in Listing 1 are good examples. The locution is exactly what is said, and the speaker is given (in this case Matthew Taylor), whereas the proposition can be thought of as adding a bit more context from the surrounding dialogue. This primarily includes pronoun resolution as seen in the first word "she" in the locution vs. "Nancy Sherman" in the proposition.

An array of these JSON objects can then be used to create the node pairs required for the training and evaluation of the model.

3.1.2 Audio Data

The audio data first had to be downsampled from 44.1kHz to the 16kHz which is best accepted by the Wav2Vec2 transformer [20] among many others. This can easily be achieved using FFmpeg[3]. In the case of QT30, first audio had to be extracted from the video, and collapsed into a mono track before it could be downsampled, this was also easily achieved with FFmpeg.

Next, start and end times for each locution in the ar-

[2]https://corpora.aifdb.org/

[3]https://ffmpeg.org/

Listing 1 Example JSON object corresponding to a Node.

```
{
    "id": 433407,
    "locution": "Matthew Taylor : she answered questions about norms and structures by talking about beliefs and campaigns and I think beliefs are different to norms and I think campaigns are different to social structures",
    "proposition": "Nancy Sherman answered questions about norms and structures by talking about beliefs and campaigns and beliefs are different to norms and campaigns are different to social structures",
    "relations": [
        {
            "type": "CA",
            "to_node_id": 433393
        },
        {
            "type": "RA",
            "to_node_id": 433416
        }
    ],
}
```

gument graph need to be found, to allow the audio to be split per-locution (and therefore per node). This can be achieved using Connectionist Temporal Classification (CTC) [52] as exposed by PyTorch's forced alignment api[4].

In order to understand the abilities of a CTC-based forced alignment system it is of course useful to understand how the algorithms work. Simply, CTC provides the probability distribution across a set of tokens, for each timestep (known as a frame). For a forced alignment task, these tokens are typically each letter of the alphabet and a blank token. The blank token is used for frames which cannot be classified as any other token (e.g. silence).

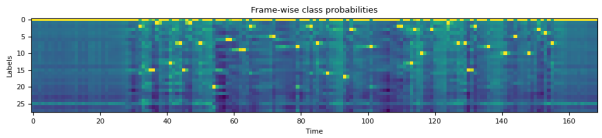

Figure 2: Example framewise probabilities.

Figure 2 (taken from the PyTorch tutorial on the subject[5]) shows an example of the framewise probability distribution across each token, token 0 here is the blank token. This distribution provides the probability (or confidence) of any particular token appearing in any given frame. Taking simply the most probable tokens provides something that looks like the following: `- i - - h h a - - - d` – where – is the blank token. That sequence describes the words 'I had', so it can be seen that the duplicates need to be removed, along with the blank tokens. This is the process that would be undertaken for Automated Speech Recognition.

When looking at forced alignment however, the process is a bit different since we already have a transcript. For forced alignment the goal is to find the most probably route through the framewise probability matrix matching the transcript. To do this a so-called trellis matrix can be generated. This represents the probabilities of remaining at the same token in the transcript, or moving on to the next one in each frame.

We are then looking for the path across the most likely transitions, $k_{(t+1,j+1)}$, where j is the current location in the transcript, and t is the current timeframe. The trellis can then be defined as in Equation 1.

$$k_{t+1,j+1} = \max\left(k_{(t,j)p(t+1,c_{j+1})}, k_{(t,j+1)}p(t+1, repeat)\right) \tag{1}$$

Where k is the trellis matrix and $p(t, c_j)$ is the probability of any token c_j appearing in frame t, effectively ref-

[4]https://pytorch.org/audio/

[5]https://pytorch.org/audio/main/tutorials/ctc_forced_alignment_api_tutorial.html

erencing the framewise probability matrix, and $repeat$ represents the blank token.

Once the trellis matrix is generated, an example of which is shown in Figure 3[6] where the yellow high-probability path is visually obvious, it can be traversed using a backtracking algorithm, starting from the last token in the transcript and following either $(c_j \rightarrow c_j)$ or $(c_j \rightarrow c_j+1)$ transitions, based on their probability, until reaching the beginning of the transcript.

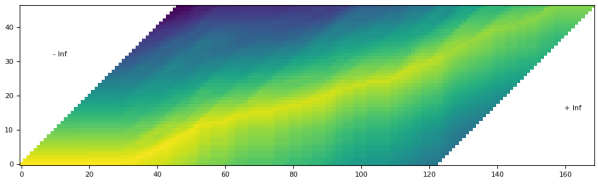

Figure 3: Example trellis matrix where yellow shows a high probability.

At this point, we have start and end frame-numbers for each token, and based on the probabilities the model has traversed, a 'confidence' score can be calculated based on the mean of the probabilities traversed, this group of three values is known as a span (in this case a token span). The token spans can be generated by using the PyTorch forced alignment API, allowing the algorithm to be easily implemented and used. Finally, the token spans can be combined into word spans based on word boundaries in the transcript. This provides start and end frame numbers for each word, along with a confidence score. The same process can be conducted later to find a locution-level span with a confidence score. The frame numbers can then be easily converted back into times in the waveform and then split into the required segment.

Initially the forced alignment of the argumentative discourse was achieved by aligning each word in the complete transcript of the episode, producing start and end times for each word. A search can then be performed through this data to find the required locution. While this technique initially produced promising results, it was not robust enough to allow for errors in the transcripts or the crosstalk common in debates. This problem can be seen in the fact that the trellis matrix only allows for a single token to appear in each timeframe, which is trivially not applicable to the real world in an argumentative context where a lot of crosstalk (multiple speakers talking over each other) exists.

To solve this problem, the PyTorch forced alignment API is able to take wildcard tokens as input, therefore, each locution can be searched for individually. To achieve this, the partial transcript used as input to the forced aligner took the following form: $\star$ $\{$`locution`$\}$ $\star$.

Using this system allows the forced aligner to work well through crosstalk (since each locution's alignments are searched for independently of all others), and qualitatively seems to be more resilient to errors. Error resilience is helped since errors are less common in the locution texts as opposed to the transcripts. Using this system also allowed for confidence scores to be collected for analysis. In this section general analysis across all corpora is performed, with corpus specific analysis in the relevant section.

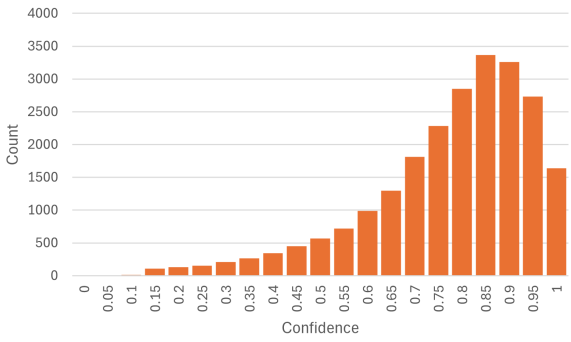

Figure 4: Confidence distribution across all corpora.

Figure 4 shows the distribution of confidence scores across both the QT30 corpus and all Moral Maze corpora. This distribution shows that the system can relatively confidently align the majority of locutions, with only approx. 8% of locutions with a confidence score less than 0.50.

In order to further analyse the performance of this system, locutions were selected at random and qualitatively analysed. Throughout this process, all locutions ap-

[6] https://pytorch.org/audio/main/tutorials/forced_alignment_tutorial.html

peared correct, however, it was very challenging to accurately determine the accuracy of the system on locutions with confidence scores < 0.2. This shows that this method of aligning locutions with their corresponding audio is accurate for the purposes of this project, as long as the confidence scores are taken into account.

The distribution of lengths for each audio clip was also analysed in order to ensure the models are being provided with enough data. The primary statistics are shown in Table 1. This data shows that the majority of locutions are shorter than 8 seconds (approximately 120,000 samples at the sampling rate of 16kHz). In total, across both corpora, there is over 24 hours of argumentative audio, out of over 36 hours of total audio processed. The relevant data for the specific corpora are detailed in the relevant section.

Table 1: Audio data for locutions across all corpora.

Quantity	Length (s)	No. of Samples
Mean	3.9	62,000
75th Percentile	5.1	81,000
90th Percentile	7.7	120,000
Maximum	31	490,000

3.1.3 Pair Creation

Finally, a set of node pairs and their relations can be generated in order to train a neural network. For related nodes this can be done trivially in that for each relation, the corresponding pair of nodes can be added to the set. When sampling unrelated nodes, however, things are more complex.

It has also been shown that how unrelated node pairs are sampled is very relevant to the model's performance [45]. For this reason, it is also useful to provide a comparison between the different methods in a multimodal context. Since a short context is defined as being within an episode, the sampling strategies are only relevant for QT30, all Moral Maze episodes are simply undersampled. The following methods are compared:

- **Undersampling (US)** is the simplest method. The set of all possible pairs is created and then randomly undersampled to the number of inference/support relations.
- **Long Context Sampling (LCS)** samples unrelated nodes such that each node comes from a different episode with the result that they are 'far apart' in the discourse, this often takes the form of a different topic and such the task is slightly easier than the other methods. This list can then be randomly undersampled to the number of inference/support relations.
- **Short Context Sampling (SCS)** samples unrelated such that each node comes from the same episode so they are 'close together' in the discourse meaning that they often involve the same topic with the result that the task is slightly harder than other methods. This set is then randomly undersampled to the number of inference/support relations.

3.2 QT30

The QT30 argument corpus [51] contains transcripts and argument annotations for 30 episodes of the BBC's Question Time, a series of televised topical debates across the United Kingdom. All episodes aired in 2020 and 2021. The corpus is split into 30 subcorpora, each spanning a single episode. This allows analysis of each episode individually, or combined as a single corpus.

Table 2: Disribution of propositional relations in QT30.

Relation Type	Count	Proportion (%)
Inference	5,761	51.4%
Conflict	947	8.5%
Rephrase	4,496	40.1%
Total	11,204	100%

Table 2 shows the distribution of each type of relation across QT30. Inference and Rephrase relations make up a total of 91.5% of the dataset, with Conflict relations being significantly less common, only making up 8.5% of the dataset. It is obvious that this is an unbalanced dataset, which will have to be considered during training.

Table 3 shows the mean and standard deviation of the confidence scores across each of the QT30 subcorpora. The lowest two mean scores are shown in bold. Manu-

Table 3: Mean confidence scores (μ) and standard deviation of confidence scores (σ) across each QT30 subcorpus.

Corpus Name	μ	σ
28May2020	0.76	0.16
4June2020	0.72	0.17
18June2020	0.76	0.16
30July2020	0.75	0.16
2September2020	0.78	0.15
22October2020	0.76	0.16
5November2020	0.77	0.17
19November2020	0.74	0.19
10December2020	0.77	0.16
14January2021	0.74	0.17
28January2021	0.70	0.17
18February2021	0.75	0.16
4March2021	0.76	0.16
18March2021	0.75	0.17
15April2021	0.75	0.15
29April2021	0.70	0.19
20May2021	0.76	0.17
27May2021	0.79	0.15
10June2021	0.75	0.17
24June2021	0.74	0.17
8July2021	0.72	0.17
22July2021	**0.44**	**0.28**
5August2021	0.76	0.17
19August2021	0.77	0.17
2September2021	0.78	0.16
16September2021	0.77	0.16
30September2021	**0.24**	**0.08**
14October2021	0.75	0.19
28October2021	0.75	0.17
11November2021	0.78	0.16
QT30	0.75	0.17

ally analysing samples in these episodes indicates a high error rate in the alignment of locutions. Because of this high error rate, it was decided to exclude these episodes from the corpus used for training. The excluded episodes are: 22July2021 and 30September2021. The rest of the episodes from QT30 will form its multimodal subcorpus (QT30-MM).

Table 4: Disribution of propositional relations in QT30-MM.

Relation Type	Count	Proportion (%)
Inference	5,740	51%
Conflict	937	8.4%
Rephrase	4,479	40.6%
Total	11,156	100%

Similarly to the complete QT30 corpus, Inference and Rephrase make up the vast majority of the QT30-MM dataset, with the proportion of Conflict relations decreasing to 8.4%.

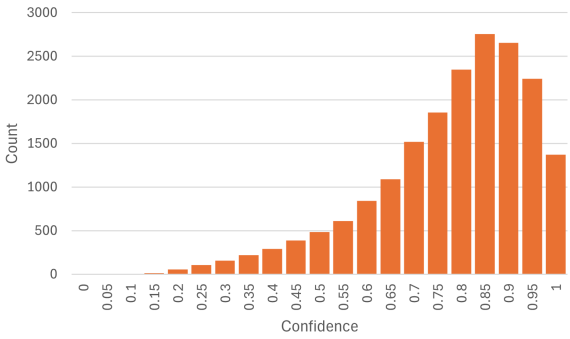

Figure 5: Confidence distribution across QT30-MM.

As can be seen in Figure 5 the distribution of confidence scores closely matches that shown in Figure 4. This still indicates that the audio alignments are calculated with high accuracy.

Table 5: Distribution of propositional relations after sampling non-related nodes.

Relation Type	Count	Proportion (%)
None	5,470	34%
Inference	5,470	34%
Conflict	937	6%
Rephrase	4,479	27%
Total	16,896	100%

Table 6 shows the statistics for the audio part of QT30-MM. Generally the values are very similar to those

shown in Table 1. The QT30-MM dataset contains almost 20 hours of argumentative audio taken from approximately 29.5 hours of total audio.

Table 6: Audio data for locutions across QT30-MM.

Quantity	Length (s)	No. of Samples
Mean	3.8	60,000
75th Percentile	5.0	79,000
90th Percentile	7.5	120,000
Maximum	30	470,000

3.3 Moral Maze

Similar to Question Time, the BBC's Moral Maze is a series of radio broadcast debates, with each episode focusing on a certain topic. Nine different Moral Maze episodes have been AIF annotated and made available on AIFdb. It is therefore these nine episodes, released from 2012 to 2019, which this project considers. Each episode focuses on a very different domain which allows for a robust, cross-domain analysis of any models trained on another corpus (e.g. QT30). The Moral Maze corpus contains data from nine different episodes: Banking (B), Empire (E), Money (M), Problem (P), Syria (S), Green Belt (G), D-Day (D), Hypocrisy (H) and Welfare (W). Each episode consists of a debate focusing on a different topic, and hence has a different distribution of classes.

Table 7 shows the distribution of propositional relations across the Moral Maze corpus, after non-related pairs have been sampled. Comparing the corpus to QT30, a significantly lower proportion of the corpus is made up of Rephrase relations. It is possible that the differing formats of the debates has an impact here.

In Table 8 the mean and standard deviation of audio alignemnt confidence scores are compared across subcorpora. Generally the results match what is expected and are similar to those in QT30, the system does achieve unusually low scores when considering the D-Day subcorpus, the reason for this is unclear, however, manually analysing both random and low-confidence samples indicates they are generally correct and so the subcorpus can be used for the cross-domain evaluation.

Figure 6 shows the distribution of audio alignment confidence scores across all Moral Maze episodes. This follows the expected pattern as shown in Figures 4 and 5. The secondary peak around a confidence of 0.10 is caused by the D-Day subcorpus.

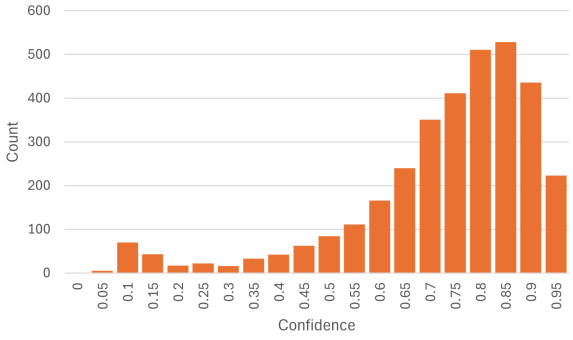

Figure 6: Confidence distribution across all Moral Maze subcorpora.

Table 9 shows the statistics for the audio part of the combined Moral Maze corpus. Generally the locutions seem to be a bit longer in the Moral Maze when compared to QT30. The Moral Maze combined corpus contains almost 5 hours of argumentative audio taken from approximately 6.5 hours of total audio.

4 Models

This section descries the model architectures that are evaluated in this project. Since the models will be aimed at a sequence pair classification task, there is a distinction between when data from each sequence is combined (which can be termed sequence fusion) and when data from each modality is combined (which can be termed multimodal fusion). The following subsections define the different approaches evaluated for each of these stages.

In previous work it has been shown that the RoBERTa models perform better than other pretrained transformers on the task of ARI [35]. To encode the text data, the RoBERTa-base model is used, with 12 encoder layers, each with 12 attention heads and a hidden size of

Table 7: Distribution of propositional relations across the Moral Maze corpus.

Relation Type	None	Inference	Conflict	Rephrase	Total
B	132 (45%)	132 (45%)	24 (8%)	3 (1%)	291
E	151 (41%)	151 (41%)	39 (11%)	25 (7%)	366
M	255 (43%)	255 (43%)	29 (5%)	58 (10%)	597
P	236 (42%)	236 (42%)	40 (7%)	45 (8%)	557
S	181 (42%)	181 (42%)	63 (14%)	10 (2%)	435
G	301 (41%)	301 (41%)	46 (6%)	93 (13%)	741
D	72 (40%)	72 (40%)	7 (4%)	28 (16%)	179
H	207 (43%)	207 (43%)	23 (5%)	43 (9%)	480
W	211 (40%)	211 (40%)	59 (11%)	43 (8%)	524
Total	1,746 (42%)	1,746 (42%)	330 (8%)	348 (8%)	4,170

Table 8: Mean confidence scores (μ) and Standard Deviation of confidence scores (σ) across Moral Maze subcorpora.

Subcorpus	μ	σ
B	0.79	0.14
E	0.76	0.15
M	0.78	0.14
P	0.80	0.15
S	0.74	0.16
G	0.80	0.14
D	**0.50**	**0.34**
H	0.74	0.16
W	0.75	0.15

768, resulting in 110M total parameters[7]. Wav2Vec 2.0 is also used in many audio processing tasks [31], [53] and therefore is used to encode the audio data. To ensure both models output the same hidden size, the wav2vec2-base model is used with 95M total parameters[8]. The wav2vec2 model used is fine-tuned on 960 hours of librispeech for automatic speech recognition.

The classification head used for most models is a simple linear projection from the hidden vector down to the required number of classes. The best performing model on ARI as proposed in MAMKit [31] (MM-RoBERTa) is also evaluated. Their model uses the fusion architecture described in Figure 8 but with a 3 layer Multilayer Perceptron (MLP) model as the classification head. They also only train the classification head, without training the text or audio encoders.

[7]https://huggingface.co/FacebookAI/roberta-base

[8]https://huggingface.co/facebook/wav2vec2-base-960h

Table 9: Audio data for locutions across Moral Maze.

Quantity	Length (s)	No. of Samples
Mean	4.4	70,000
75th Percentile	5.8	94,000
90th Percentile	9.0	140,000
Maximum	31	490,000

4.1 Sequence Fusion

In most text-processing approaches data from each sequence is combined at the text level using special tokens defined in the encoder's tokeniser [36], [38], [39]. For this project, this approach is termed early sequence fusion. To achieve this, the input sequences can be delimiter by a separator token, and the entire sequence wrapped in the start of sequence (SOS) and end of sequence (EOS) tokens. An example using the RoBERTa tokeniser takes the following form: `<s> [sequence 1] </s> [sequence 2] </s>`. Here the `<s>` token corresponds to the SOS, and `</s>` does the job of both the separator token and the EOS token.

As far as was found, there is no existing literature on how early sequence fusion could function for audio models. Therefore, it was decided to deliniate each audio sequence by a certain amount of silence. The exact amount

of silence could be adjusted as a training hyperparameter and was eventually set to 5 seconds.

Figure 7 shows an example of a model architecture using this late fusion technique, text related steps are shown in purple and audio related steps are shown in green. RoBERTa and Wav2Vec2 are simply used as examples and could be substituted for other models.

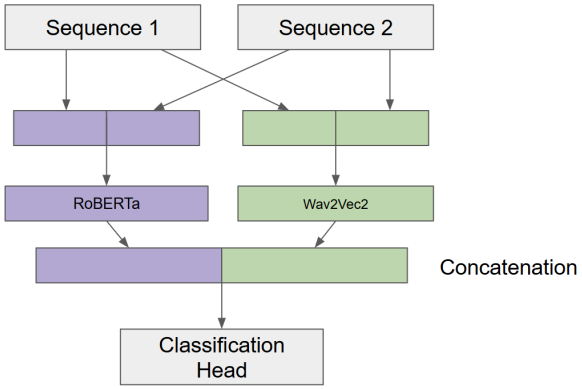

Figure 7: Model diagram with early sequence and late multimodal fusion.

Mestre *et al.* [30] and Mancini *et al.* [31] approach the problem differently. They first put each sequence through the text encoder independently, before fusing the outputs and feeding the combined encodings into the classification head. While concatenation is the only fusion method examined for this sequence fusion technique, others (such as an element-wise product or cross-attention) could be used. This approach extends much more easily to the audio modality, since the audio encodings can be combined in the same way as the text encodings. This approach to fusing the data in each sequence can be termed late sequence fusion. Figure 8 shows an example of a model architecture using late sequence fusion, text processing steps and data are shown in purple and audio-related steps are shown in green.

4.2 Multimodal Fusion

Multimodal fusion decribes the method by which the text and audio data is combined. As detailed in Section 2.2, fusion techniques can be split into two major categories:

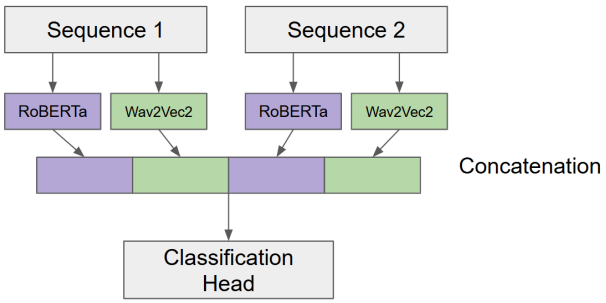

Figure 8: Model diagram with late concatenation sequence and multimodal fusion.

early and late fusion. This project only evaluates late fusion techniques due to their ease of development and the applicability of pre-trained models. The following techniques are evaluated:

- **Concatenation** where the pooled encodings for each modality are simply concatenated before being fed into the classification head.
- **Elementwise-product** (otherwise known as a Hadamard product) takes the product of each element in the pooled encodings for each modality.
- **Crossmodal Attention** (CA) is similar to the self-attention mechanism found in transformers, however, the queries are taken from a different modality when compared to the keys and values. To compute crossmodal attention features, the query and key matrices are multiplied and then put through a softmax. This is then multiplied with the value matrix and the result can then be pooled using an arithmetic mean. For the purposes of this project, the CA module is labelled based on the modality from which the query matrix is derived (e.g. a `CA_Text` module derives the query matrix from the text encodings and the key and value matrices from the audio encodings). This is shown in Figure 9.

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{2}$$

Listing 2 shows how a crossmodal attention mechanism can be implemented into a PyTorch module. The code is

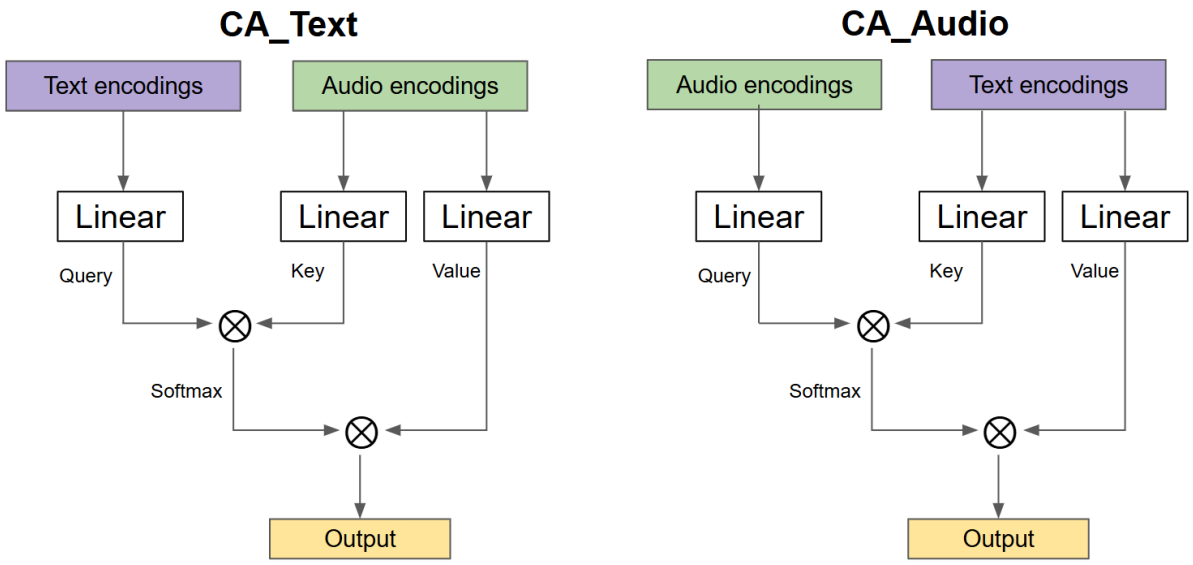

Figure 9: Crossmodal attention system with both text and audio queries. $\otimes$ is used to denote matrix multiplication.

contained within the forward method of a PyTorch module, where `self.query_proj`, `self.key_proj` and `self.value_proj` are the linear projections for the queries, keys and values respectively. `torch.bmm` refers to a matrix multiplication and `F.softmax` is a mathematical function to ensure all values (across the requested dimension) lie in the range $[0, 1]$ and sum to 1, the softmax function for a vector $\mathbf{x}$ is given in Equation 2.

5 Results

5.1 Experimental Setup

To provide a comparable set of results, all experiments were run using the same hyperparameters. Each model was trained on a single Nvidia RTX 4070 Super for 15 epochs with a batch size of 32 using a weighted cross-entropy loss and the AdamW optimiser [54] initialised with a learning rate of 10^{-5}, a linear learning rate scheduler and 10% of training used as warm-up steps. The cross-entropy weights were calculated as in Equation 3, where $\mathbf{c}$ is a vector containing the number of samples in each class, and $\mathbf{w}$ is a vector containing the relevant cross-entropy weight.

$$w_i = \frac{\max(\mathbf{c})}{c_i} \tag{3}$$

In order to evaluate the models, the following metrics are reported: macro-averaged F1 score, precision and recall. These are all described for each class in Equations 4, 5 and 6 where TP is the number of true positives, FP is the number of false positives and FN is the number of false negatives. The arithmetic mean can then be taken for each class to provide a holistic overview of the model's performance.

$$F1 = \frac{2TP}{2TP + FP + FN} \tag{4}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{6}$$

Listing 2 PyTorch forward method for a crossmodal attention mechanism.

```
# project query features
queries = self.query_proj(query_modality)

# project kv features
keys = self.key_proj(kv_modality)
values = self.value_proj(kv_modality)

# compute attention scores
attn_scores = torch.bmm(queries, keys.transpose(1, 2))
attn_probs = F.softmax(attn_scores, dim=-1)

# compute and return cross modal features
return torch.bmm(attn_probs, values)
```

Generally the macro-averaged F1 score is the standard to evaluate a multi-class classification problem, including ARI systems [31], [35], [45], where only a single metric is reported, it will be a macro-averaged F1 score for this reason.

The QT30-MM dataset is split into three splits: train, validation and test. 70% of the data is allocated for training, 10% for validation and the remaining 20% for testing. The model is evaluated on the validation split after every training epoch, the best performing model, based on macro-F1, is then chosen to be tested on the testing split, the metrics are then calculated and reported in the following sections.

Each model is then evaluated on the complete dataset for each Moral Maze episode (Banking, Empire, Money, Problem, Syria, Green Belt and D-Day) and each metric calculated to provide an overview of the cross-domain performance of the model.

In order to evaluate the different methods to sample unrelated arguments as described in Section 3.1.3. Models are trained on SCS, US and LCS, the validation dataset is sampled identically to the training set, and tested, both in-domain and cross-domain on SCS. This is used because of its description as a more realistic problem [45]. All code used for the experiments can be found on GitHub[9].

5.2 In-Domain

Results are reported for both the 3-class problem (considering support, attack and no relation) and the 4-class problem (considering RA, CA, MA and NO). First results are considered when evaluating in-domain, i.e. on the test set of the QT30-MM dataset. Only macro-f1 scores are reported here, precision and recall scores are also reported in Appendix A.

5.2.1 The 4-Class Problem

Table 10 shows the macro-f1 scores of each model using RoBERTa-base as the text encoder and Wav2Vec2-base as the audio encoder. All that is shown are the results for models performing early sequence fusion across different NO-sampling strategies. In the 4-class problem it does not seem that the addition of acoustic features makes much if any difference to the performance of the model on the ARI task. This result has also been found by others [30]. However, a more in-depth discussion of the results may still yield some understanding in their limitations and how they could be improved. In order to do this the Text-Only model trained on SCS is explained in detail, however, the conclusions were found to hold on other models.

Table 10: Macro-F1 scores for early sequence fusion models on the 4-class problem. Highest results are shown in bold.

Model	**SCS**	**LCS**	**US**
Text-Only	.58	**.59**	**.59**
Audio-Only	.43	.41	.20
Concatenation	.58	.57	.58
Product	.56	.57	.58
CA Text	.57	.46	.57
CA Audio	.58	.57	.57
Random	.22	.23	.24
Majority	.14	.14	.14

[9] https://github.com/Syn-Tax/cross-domain-am

A good place to begin here is by analysing the class F1 distribution, here F1 scores are reported for each of the four classes (NO, RA, CA and MA). As can be seen in Table 11 the model performs significantly worse when shown a conflict relation as opposed to the other possible classes. It is possible that this is due to the significant class imbalance present in almost all ARI datasets as discussed in Section 3.

Table 11: Class F1 distribution for text-only SCS model on test split.

NO	RA	CA	MA
.76	.60	.30	.66

A further analysis can be conducted by looking at the confusion matrix generated as shown in Figure 10. The ideal confusion matrix shows a diagonal line, in this case from the top left down to the bottom right of the matrix, and can be used to determine which classes the model struggles to distinguish. For ARI, it is generally expected that the model is able to distinguish CA from other classes, while RA and MA are often confused with each other and sometimes with NO. This follows from the difficulties that human annotators have when determining the different relations [34]. However, this is not what Figure 10 shows, instead the model is generally confusing most classes, most notable is the underprediction of the CA class. This implies the model is simply predicting the majority classes which is a well known and well studied problem in all classification problems involving unbalanced data [55]. Typically such problems are relatively simple to solve, often using either weighted loss functions (as is explained in Section 5.1) or some form of data augmentation or manipulation technique. During this project, primarily random resampling was experimented with. Random resampling generally involves a combination (or only one) of oversampling minority classes (e.g. randomly duplicating samples labelled as CA) or undersampling majority classes (e.g. randomly discarding samples labelled as RA or NO). Often this has been shown to be the best technique for solving class-imbalance problems, despite the rise of other resampling techniques (such as SMOTE) [56]. Unfortunately in this case no resampling distribution could be found to meaningfully improve the model performance.

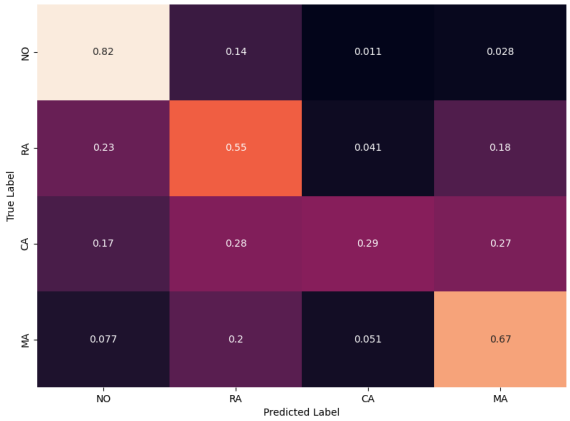

Figure 10: Confusion matrix showing true and predicted labels for the text-only SCS model.

5.2.2 The 3-Class Problem

Table 12 shows the macro-f1 scores across each NO-sampling strategy for the 3-class problem using RoBERTa-base as the text encoder and Wav2Vec2-base as the audio encoder. Similarly to the 4-class problem, the addition of acoustic features to not seem to make an appreciable difference to the performance of the model. However, it is still useful to discuss the results in more detail, again using the text-only model trained on SCS.

Table 12: Macro-F1 scores for early sequence fusion models on the 3-class problem. Highest results are shown in bold.

Model	SCS	LCS	US
Text-Only	.62	.59	.61
Audio-Only	.21	.54	.22
Concatenation	.61	.61	.60
Product	.58	.61	.62
CA Text	.53	.53	.52
CA Audio	.61	.62	**.63**
Random	.28	.30	.29
Majority	.22	.22	.22

The class F1 distribution for the 3-class problem is

shown in Table 13. Similarly to the 4-class F1 distribution, the Attack class appears to be significantly harder to prodict than the other classes. Similarly to the 4-class problem it can be hypothesised that this is due to the class imbalance in the dataset. What is interesting, is the fact that the class F1 scores for the Attack/CA relations have dropped when compared with the 4-class results and the scores for non-attack/CA relations have risen. It is possible that this is simply due to the change in the number of classes, however, it can also be considered that the 3-class dataset is more heavily unbalanced against the Attack relation which could also have this effect.

Table 13: Class F1 distribution for text-only SCS model on test split.

None	Support	Attack
.82	.78	.26

The confusion matrix for the 3-class problem, as shown in Figure 11 can also be discussed. Here the underprediction of the attack class and the overprediction of the majority class. When compared to the 4-class confusion matrix (Figure 10) it appears that the model is effectively combining incorrect predictions between samples labelled inference and rephrase. This result could imply that the model's learning is approximately equivalent between the 3- and 4-class approaches, although it should be noted that more investigation is required to prove this.

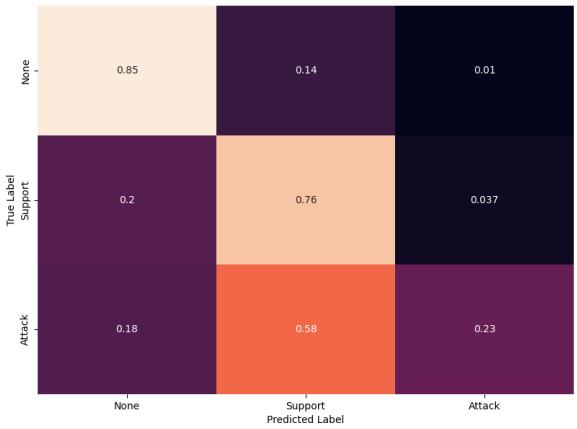

Figure 11: Confusion matrix showing true and predicted labels for the text-only SCS model.

5.3 Cross-Domain

5.3.1 The 4-Class Problem

5.3.2 The 3-Class Problem

6 Limitations

7 Conclusions

8 Future Work

References

[1] C. L. Hamblin, *Fallacies*, First Edition. London: Methuen young books, 1970.

[2] J. R. Searle, *Speech Acts: An Essay in the Philosophy of Language*. Cambridge: Cambridge University Press, 1969. doi: 10.1017/CBO9781139173438.

[3] W. C. Mann and S. A. Thompson, "Rhetorical Structure Theory: Toward a functional theory of text organization," *Text - Interdisciplinary Journal for the Study of Discourse*, vol. 8, no. 3, pp. 243–281, Jan. 1988, doi: 10.1515/text.1.1988.8.3.243.

Table 14: Cross-Domain macro-averaged F1 scores on 4-class SCS trained models. Best scores in each column are shown in bold.

Model	B	E	M	P	S	G	H	W	Mean
Text-Only	.44	.46	**.48**	.42	.43	.50	**.51**	.41	.46
Audio-Only	.34	.37	.39	.38	.33	.38	.40	.37	.37
Concatenation	.43	.43	.45	**.45**	.45	.49	.45	.43	.45
Product	.41	.44	.42	.42	**.46**	**.52**	.46	.43	.45
CA Text	.40	.44	.44	.40	.42	.49	.43	.44	.43
CA Audio	**.45**	**.49**	.44	.42	.45	.53	.48	**.46**	**.47**
Random	.19	.23	.20	.19	.22	.25	.21	.24	.22
Majority	.16	.15	.15	.15	.15	.14	.15	.14	.15

Table 15: Cross-Domain macro-averaged F1 scores on 3-class SCS trained models. Best scores in each column are shown in bold.

Model	B	E	M	P	S	G	H	W	Mean
Text-Only	.54	.47	.50	**.50**	.58	.55	.63	.51	**.54**
Audio-Only	.21	.20	.22	.21	.20	.21	.22	.21	.21
Concatenation	.58	**.49**	.47	**.50**	.57	**.57**	.59	.49	**.54**
Product	.51	.45	.44	.47	.53	.54	**.64**	.47	.51
CA Text	.43	.40	.43	.41	.44	.46	.47	.44	.44
CA Audio	**.59**	.44	**.52**	.48	**.59**	.54	.60	**.52**	**.54**
Random	.27	.32	.30	.27	.30	.29	.29	.33	.30
Majority	.21	.21	.22	.21	.20	.21	.22	.20	.21

[4] C. Reed and K. Budzynska, "How dialogues create arguments," 2011.

[5] C. Reed, "A Quick Start Guide to Inference Anchoring Theory (IAT)." Sep. 2017.

[6] D. Walton, C. Reed, and F. Macagno, *Argumentation Schemes*. Cambridge: Cambridge University Press, 2008. doi: 10.1017/CBO9780511802034.

[7] D. Walton, "Argumentation Theory: A Very Short Introduction," in *Argumentation in Artificial Intelligence*, G. Simari and I. Rahwan, Eds. Boston, MA: Springer US, 2009, pp. 1–22. doi: 10.1007/978-0-387-98197-0_1.

[8] K. Budzynska, M. Janier, C. Reed, P. Saint-Dizier, M. Stede, and O. Yakorska, "A Model for Processing Illocutionary Structures and Argumentation in Debates," in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, May 2014, pp. 917–924.

[9] C. Reed and G. Rowe, "Araucaria: Software for argument analysis, diagramming and representation," *International Journal on Artificial Intelligence Tools*, vol. 13, no. 4, pp. 961–979, Dec. 2004, doi: 10.1142/S0218213004001922.

[10] C. Chesñevar *et al.*, "Towards an argument interchange format," *The Knowledge Engineering Review*, vol. 21, no. 4, pp. 293–316, Dec. 2006, doi: 10.1017/S0269888906001044.

[11] C. Reed, J. Devereux, S. Wells, and G. Rowe, "AIF+: Dialogue in the Argument Interchange Format."

[12] I. Rahwan, F. Zablith, and C. Reed, "Laying the foundations for a World Wide Argument Web," *Artificial Intelligence*, vol. 171, no. 10–15, pp. 897–921, Jul. 2007, doi: 10.1016/j.artint.2007.04.015.

[13] J. Lawrence, F. Bex, C. Reed, and M. Snaith, "AIFdb: Infrastructure for the Argument Web," in *Computational Models of Argument*, IOS Press, 2012, pp. 515–516. doi: 10.3233/978-1-61499-111-3-515.

[14] A. Vaswani *et al.*, "Attention Is All You Need," p. 11, 2017.

[15] M. E. Peters *et al.*, "Deep contextualized word representations." arXiv, Mar. 2018. doi: 10.48550/arXiv.1802.05365.

[16] A. M. Dai and Q. V. Le, "Semi-supervised Sequence Learning." arXiv, Nov. 2015. doi: 10.48550/arXiv.1511.01432.

[17] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv, May 2019. doi: 10.48550/arXiv.1810.04805.

[18] Y. Liu *et al.*, "RoBERTa: A Robustly Optimized BERT Pretraining Approach." arXiv, Jul. 2019. doi: 10.48550/arXiv.1907.11692.

[19] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "Wav2vec: Unsupervised Pre-training for Speech Recognition." arXiv, Sep. 2019. Accessed: Oct. 17, 2024. [Online]. Available: https://arxiv.org/abs/1904.05862

[20] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "Wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations." arXiv, Oct. 2020. Accessed: Oct. 16, 2024. [Online]. Available: https://arxiv.org/abs/2006.11477

[21] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, "HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units." arXiv, Jun. 2021. doi: 10.48550/arXiv.2106.07447.

[22] OpenAI *et al.*, "GPT-4 Technical Report." arXiv, Mar. 2024. doi: 10.48550/arXiv.2303.08774.

[23] H. Touvron *et al.*, "LLaMA: Open and Efficient Foundation Language Models." arXiv, Feb. 2023. doi: 10.48550/arXiv.2302.13971.

[24] T. B. Brown *et al.*, "Language Models are Few-Shot Learners." arXiv, Jul. 2020. doi: 10.48550/arXiv.2005.14165.

[25] A. Sharma, A. Gupta, and M. Bilalpur, "Argumentative Stance Prediction: An Exploratory Study on Multimodality and Few-Shot Learning," in *Proceedings of the 10th Workshop on Argument Mining*, Dec. 2023, pp. 167–174. doi: 10.18653/v1/2023.argmining-1.18.

[26] J. Delbrouck *et al.*, "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*, May 2022, pp. 23–34. doi: 10.18653/v1/2022.acl-demo.3.

[27] K. Sun *et al.*, "CMAF-Net: A cross-modal attention fusion-based deep neural network for incomplete multi-modal brain tumor segmentation," *Quantitative Imaging in Medicine and Surgery*, vol. 14, no. 7, pp. 4579–4604, Jul. 2024, doi: 10.21037/qims-24-9.

[28] E. Toto, M. Tlachac, and E. A. Rundensteiner, "AudiBERT: A Deep Transfer Learning Multimodal Classification Framework for Depression Screening," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, Oct. 2021, pp. 4145–4154. doi: 10.1145/3459637.3481895.

[29] Y.-H. H. Tsai, S. Bai, P. P. Liang, J. Z. Kolter, L.-P. Morency, and R. Salakhutdinov, "Multimodal Transformer for Unaligned Multimodal Language Sequences," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Jul. 2019, pp. 6558–6569. doi: 10.18653/v1/P19-1656.

[30] R. Mestre, R. Milicin, S. E. Middleton, M. Ryan, J. Zhu, and T. J. Norman, "M-Arg: Multimodal Argument Mining Dataset for Political Debates with Audio and Transcripts," in *Proceedings of the 8th Workshop on Argument Mining*, Nov. 2021, pp. 78–88. doi: 10.18653/v1/2021.argmining-1.8.

[31] E. Mancini, F. Ruggeri, S. Colamonaco, A. Zecca, S. Marro, and P. Torroni, "MAMKit: A Comprehensive Multimodal Argument Mining Toolkit," in *Proceedings of the 11th Workshop on Argument Mining (ArgMining 2024)*, Aug. 2024, pp. 69–82. doi: 10.18653/v1/2024.argmining-1.7.

[32] V. Rajan, A. Brutti, and A. Cavallaro, "Is Cross-Attention Preferable to Self-Attention for Multi-Modal Emotion Recognition?" arXiv, Feb. 2022. doi: 10.48550/arXiv.2202.09263.

[33] L. Ye, M. Rochan, Z. Liu, and Y. Wang, "Cross-Modal Self-Attention Network for Referring Image Segmentation." arXiv, Apr. 2019. doi: 10.48550/arXiv.1904.04745.

[34] J. Lawrence and C. Reed, "Argument Mining: A Survey," *Computational Linguistics*, vol. 45, no. 4, pp. 765–818, Jan. 2020, doi: 10.1162/coli_a_00364.

[35] R. Ruiz-Dolz, J. Alemany, S. M. H. Barberá, and A. García-Fornes, "Transformer-Based Models for Automatic Identification of Argument Relations: A Cross-Domain Evaluation," *IEEE Intelligent Systems*, vol. 36, no. 6, pp. 62–70, Nov. 2021, doi: 10.1109/MIS.2021.3073993.

[36] D. Gemechu, R. Ruiz-Dolz, and C. Reed, "ARIES: A General Benchmark for Argument Relation Identification," in *Proceedings of the 11th Workshop on Argument Mining (ArgMining 2024)*, Aug. 2024, pp. 1–14. doi: 10.18653/v1/2024.argmining-1.1.

[37] D. Gorur, A. Rago, and F. Toni, "Can Large Language Models perform Relation-based Argument Mining?" arXiv, Feb. 2024. doi: 10.48550/arXiv.2402.11243.

[38] Y. Wu, Y. Zhou, B. Xu, W. Wang, and Y. Song, "KnowComp at DialAM-2024: Fine-tuning Pre-trained Language Models for Dialogical Argument Mining with Inference Anchoring Theory," in *Proceedings of the 11th Workshop on Argument Mining (ArgMining 2024)*, Aug. 2024, pp. 103–109. doi: 10.18653/v1/2024.argmining-1.10.

[39] Z. Zheng, Z. Wang, Q. Zong, and Y. Song, "KNOWCOMP POKEMON Team at DialAM-2024: A Two-Stage Pipeline for Detecting Relations in Dialogue Argument Mining," in *Proceedings of the 11th Workshop on Argument Mining (ArgMining 2024)*, Aug. 2024, pp. 110–118. doi: 10.18653/v1/2024.argmining-1.11.

[40] S. Eger, J. Daxenberger, and I. Gurevych, "Neural End-to-End Learning for Computational Argumentation Mining," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Jul. 2017, pp. 11–22. doi: 10.18653/v1/P17-1002.

[41] S. Haddadan, E. Cabrio, and S. Villata, "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*, Jul. 2019, pp. 4684–4690. doi: 10.18653/v1/P19-1463.

[42] C. Stab, T. Miller, B. Schiller, P. Rai, and I. Gurevych, "Cross-topic Argument Mining from Heterogeneous Sources," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Oct. 2018, pp. 3664–3674. doi: 10.18653/v1/D18-1402.

[43] K. Al-Khatib, H. Wachsmuth, M. Hagen, J. Köhler, and B. Stein, "Cross-Domain Mining of Argumentative Text through Distant Supervision," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Jun. 2016, pp. 1395–1404. doi: 10.18653/v1/N16-1165.

[44] S. Eger, J. Daxenberger, C. Stab, and I. Gurevych, "Cross-lingual Argumentation Mining: Machine Translation (and a bit of Projection) is All You Need!" in *Proceedings of the 27th International Conference on Computational Linguistics*, Aug. 2018, pp. 831–844.

[45] R. Ruiz-Dolz, D. Gemechu, Z. Kikteva, and C. Reed, "Looking at the Unseen: Effective Sampling of Non-Related Propositions for Argument Mining," in *Proceedings of the 31st International Conference on Computational Linguistics*, Jan. 2025, pp. 2131–2143.

[46] Z. Liu, M. Guo, Y. Dai, and D. Litman, "ImageArg: A Multi-modal Tweet Dataset for Image Persuasiveness Mining," in *Proceedings of the 9th Workshop on Argument Mining*, Oct. 2022, pp. 1–18.

[47] Q. Zong *et al.*, "TILFA: A Unified Framework for Text, Image, and Layout Fusion in Argument Mining," in *Proceedings of the 10th Workshop on Argument Mining*, Dec. 2023, pp. 139–147. doi: 10.18653/v1/2023.argmining-1.14.

[48] Z. Liu, M. Elaraby, Y. Zhong, and D. Litman, "Overview of ImageArg-2023: The First Shared Task in Multimodal Argument Mining," in *Proceedings of the 10th Workshop on Argument Mining*, Dec. 2023, pp. 120–132. doi: 10.18653/v1/2023.argmining-1.12.

[49] E. Mancini, F. Ruggeri, A. Galassi, and P. Torroni, "Multimodal Argument Mining: A Case Study in Political Debates," in *Proceedings of the 9th Workshop on Argument Mining*, Oct. 2022, pp. 158–170.

[50] R. Ruiz-Dolz and J. Iranzo-Sánchez, "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*, Dec. 2023, pp. 2071–2077. doi: 10.18653/v1/2023.emnlp-main.128.

[51] A. Hautli-Janisz, Z. Kikteva, W. Siskou, K. Gorska, R. Becker, and C. Reed, "QT30: A Corpus of Argument and Conflict in Broadcast Debate," in *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, Jun. 2022, pp. 3291–3300.

[52] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning - ICML '06*, 2006, pp. 369–376. doi: 10.1145/1143844.1143891.

[53] E. Mancini, F. Ruggeri, and P. Torroni, "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)*, Mar. 2024, pp. 170–178.

[54] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization." arXiv, Jan. 2019. doi: 10.48550/arXiv.1711.05101.

[55] N. Junsomboon and T. Phienthrakul, "Combining Over-Sampling and Under-Sampling Techniques for Imbalance Dataset," in *Proceedings of the 9th International Conference on Machine Learning and Computing*, Feb. 2017, pp. 243–247. doi: 10.1145/3055635.3056643.

[56] R. Mohammed, J. Rawashdeh, and M. Abdullah, "Machine Learning with Oversampling and Undersampling Techniques: Overview Study and Experimental Results," in *2020 11th International Conference on Information and Communication Systems (ICICS)*, Apr. 2020, pp. 243–248. doi: 10.1109/ICICS49469.2020.239556.

A Results

Table 16: In-domain results when testing different model architectures on the 4-class problem. F1: Macro-averaged F1, P: precision, R: recall. Highest scores in each column are shown in bold.

Fusion Methods		SCS			LCS			US		
Sequence	Multimodal	F1	P	R	F1	P	R	F1	P	R
Text Only										
Early Late	- -	**.58**	**.58**	**.58**	**.59**	**.59**	**.59**	**.59**	**.59**	**.59**
Audio Only										
Early Late	- -	.43	.48	.44	.41	.41	.42	.20	.31	.26
Multimodal										
Early	Concatenation	**.58**	**.58**	**.58**	.57	.57	.57	.58	.58	.57
	Product	.56	.56	.57	.57	.57	.57	.58	.58	**.59**
	CA Text	.57	.56	**.58**	.46	.46	.48	.57	.56	.57
	CA Audio	**.58**	**.58**	**.58**	.57	**.59**	.56	.57	.58	.57
Late	Concatenation	.38	.40	.38	.36	.39	.36	.31	.31	.32
Baselines										
Random		.22	.24	.23	.23	.25	.24	.24	.25	.26
Majority		.14	.09	.25	.14	.09	.25	.14	.09	.25

Table 17: In-domain results when testing different model architectures on the 3-class problem. F1: Macro-averaged F1, P: precision, R: recall. Highest scoring models (based on Macro-F1) are shown in bold.

3-class										
Model		SCS			LCS			US		
Sequence	Multimodal	F1	P	R	F1	P	R	F1	P	R
Text Only										
Early Late	- -	**.62**	**.63**	**.62**	.59	**.63**	.58	.61	.63	.60
Audio Only										
Early Late	- -	.21	.16	.33	.54	.54	.55	.22	.16	.33
Multimodal										
Early	Concatenation	.61	.62	.60	.61	.61	.61	.60	.61	.59
	Product	.58	.60	.57	.61	.62	.61	.62	**.64**	.61
	CA Text	.53	.52	.54	.53	.52	.55	.52	.51	.53
	CA Audio	.61	.62	.60	**.62**	**.63**	**.62**	**.63**	.62	**.64**
Late	Concatenation									
Baselines										
Random		.28	.33	.33	.30	.34	.36	.24	.25	.26
Majority		.22	.16	.33	.22	.16	.33	.22	.16	.33