Graphical Summarisation of Argumentative Text

Jonathan Clayton

Doctor of Philosophy

The University of Sheffield

February 2026

Abstract

This work investigates the summarisation of argumentative text. Our main focus is on the generation of graphical summaries from dialogical argumentative texts such as online news comment sections.

We develop a novel type of graph structure to summarise argument, which we refer to as Argument Summary Graphs or ASGs. Our contributions are twofold. Firstly, we develop two new data resources to investigate the generation of ASGs, the Debatabase-ASG dataset (created from a curated collection of online debates), and the SENSEI-ASG dataset, developed by adding annotations to the SENSEI dataset of online news article comments.

Secondly, we investigate alternative methods for generating ASGs from both datasets using Large Language Models (LLMs). We carry out experiments on the Debatabase-ASG dataset and find that an end-to-end text-to-text method performs better than a pipeline approach. Additionally, we test how well the end-to-end approach generalises across corpora, using both of the corpora we created, and a third argument mining corpus, the Argument Annotated Essays corpus (AAEC). We find that additional fine-tuning on a monological dataset from a distinct Argument Mining task provides similar benefits to fine-tuning on a second in-genre dataset.

We also carry out two strands of work not narrowly focused on ASGs. Firstly, we investigate the prediction of Reasoning Markers, used for detection of argumentative text. We create a corpus for this task and implement multiple baselines, with the best achieving an F1 score of 0.69. Secondly, we investigate

Argument Structure Parsing: the task of extracting argumentative components and their relations from text. We fine-tune several LLMs which achieve near state-of-the-art scores in the end-to-end setting. We propose a novel method of formatting the output for this task, which we find competitive with the existing state-of-the-art approach in evaluation metrics, while being significantly faster to generate.

Acknowledgements

This work was supported by the Centre for Doctoral Training in Speech and Language Technologies (SLT) and their Applications funded by UK Research and Innovation [grant number EP/S023062/1], and by Amazon.

Thanks are firstly due to my supervisor Rob Gaizauskas, whose help has made the whole process possible. He has tirelessly helped me stay focused on the end goal and was always on hand to give feedback, despite his absurdly full schedule.

I'm also indebted to Marco Damonte, my industry supervisor, who has always had excellent advice for new angles to explore, as well as clear and incisive criticisms.

I'd also like to thank all my fellow students in CDT Cohort 2 for the convivial atmosphere, as well as Stuart and Lizzie for all their support.

I would like to thank my parents Sue and Charles as well; they deserve a lot of the credit for this thesis for all the work they have put in on me over the years, as well as tolerating my various impromptu visits over the course of my PhD career.

And finally to Lychee, my wife, I could not have got through it at all without your constant love and support.

Declaration

I, the author, confirm that the Thesis is my own work. I am aware of the University's Guidance on the Use of Unfair Means (https://www.sheffield.ac.uk/study-skills/assessment/academic-integrity/academic-integrity). This work has not been previously been presented for an award at this, or any other, university.

Contents

Appendices

Chapter 1

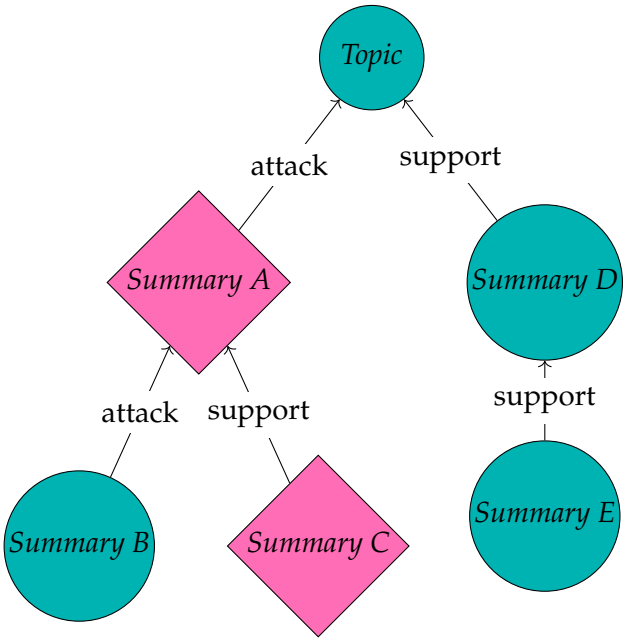
Introduction

For anyone wishing to understand a contemporary political or social debate, it is vital to understand not just the opinions that people hold, but also their reasons for doing so.

Argument Mining (AM) is a growing subfield of Natural Language Processing which addresses this task. Much work in AM extracts structures of reasoning from text, typically including argumentative components such as "premises" or "conclusions", and the relations between different argumentative components, such as "support" or "attack".

Argumentative online dialogue, in particular, is a domain which is currently of interest due to the increasing volume of online debates and the degree to which online argument is now playing a role within public discourse and shaping politics (Calderaro, 2017).

However, despite interest in this field, and a significant amount of work within AM around the summarisation of argument, there have been few works up to this point which investigate the creation of *graphical* summaries of argument. This is surprising due to the centrality of graphical representations in AM. This thesis is a contribution towards filling that gap.



Topic: Housewives should be paid for their work.

Comment A: By paying housewives for their work, you create negative stereotypes about families and women by commodifying the role of home-keeper. Paying housewives for their work re-enforces the...

Comment B: Paying housewives a wage would improve not reduce social mobility. Many women would still choose to go to university and the vast majority who do will still want to work. Paying housewives...

Comment C: The result is that women are discouraged from seeking to fulfil their own dreams by creating their own careers as they are more firmly chained to their traditional role. This is damaging to societal views...

Comment D: It is estimated that the value of a housemaker's services would be equivalent to approximately £30,000 per year. In the same way that any product or service is created...

...

Summary A: Paying housewives reduces social mobility

Summary B: It will improve women's financial freedom

Summary C: Reinforces social expectations for women

Summary D: Housewives are entitled to pay

...

Figure 1.1: An example Argument Summary Graph that could be generated from an online debate. Summaries corresponding to each node are shown below the tree for clarity. Summaries in ● support the top-level comment (Topic); summaries in ◆ argue against. Example from `idebate.net`

1.1 Definition of "Graphical Summary"

Figure 1.1 shows an example of our conception of a "graphical summary". The example is taken from Debatabase-ASG, a corpus whose generation we describe in Chapter 6.

Our definition of "graphical summary" specifically focuses on dialogue; and furthermore, summarising individual dialogic texts, in contrast to works which summarise entire text collections (for example Altemeyer et al. 2025).

As we define it, a graphical summary is a graph in which the nodes contain summaries of argumentative text units within a dialogue (such as comments or utterances), and the edges show argumentative relations between those arguments. The summary nodes can either represent one or multiple comments/ utterances.

This definition will be expanded on later in the thesis, specifically in Chapter 5, where we explain the specific type of graphical summary we propose (which we call Argument Summary Graphs or ASGs), and also how we specifically envision them being used.

1.2 Research Goals

Our general research goals are as follows:

Research Goal 1: To investigate the performance of different Natural Language Processing techniques, in particular Large Language Models, in producing graphical summaries of dialogue.

Research Goal 2: To develop resources for the generation of graphical sum-

maries of argument.

Research Goal 3: To investigate techniques for the generation of free-text summaries from graphical summaries.

In this introductory chapter, we will motivate the thesis by first analysing a sample of argumentative dialogue, which will illustrate the difficulties inherent in this domain and show why it merits further study (Section 1.3). Subsequently, we will give a chapter-by-chapter outline of the work carried out and the contributions we have made (Section 1.5).

1.3 Argument in Online Dialogue

In this section, we analyse three aspects of online argumentative dialogue, which makes it uniquely difficult to summarise. These are the following:

1. Topical diversity: online dialogues tend to have a higher degree of topical diversity compared to other text types, such as newspaper articles.
2. Nested argument: arguments can be "nested" or "recursive" — that is, arguments can develop around the premises of other arguments. This can lead to a complex structure which is difficult to summarise.
3. Online comments can sometimes display a high degree of repetition or redundancy. It is important for a summarisation system to identify these repeated components in order to produce a succinct output.

1.3.1 Topical Diversity

To illustrate the first point, we use SENSEI dataset (Barker et al., 2016), which contains newspaper articles and comments on them from the *Guardian* news website [1]. We compare the topical diversity of each article to its comments section. In order to calculate this, we first use GPT3 (Brown et al., 2020) to find the main topic of a sample of units from each text — comments in the case of a comments section, and paragraphs in the case of a news article [2]. Then for each text, we calculate a topical diversity index by dividing the number of unique topics by the total number of topics.

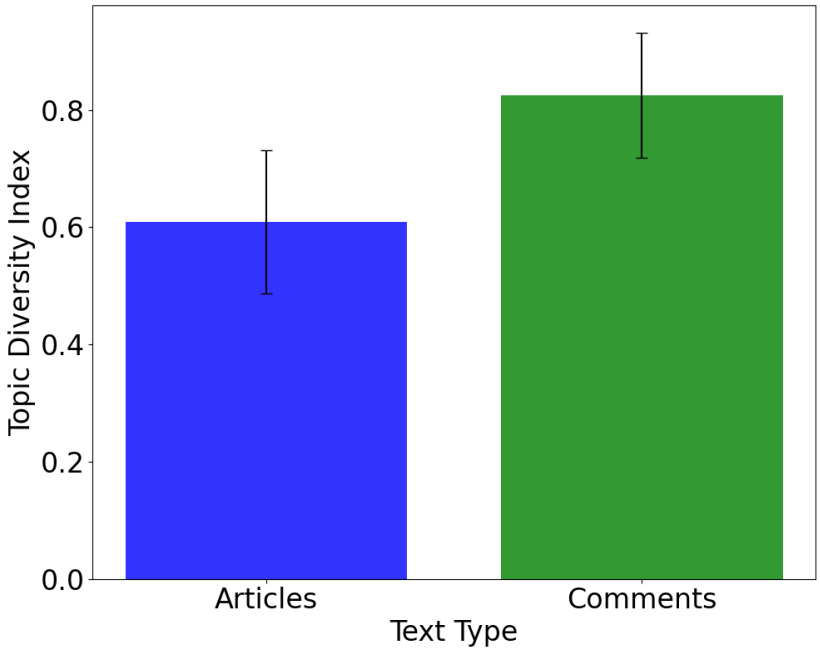


Figure 1.2: A comparison of the topical diversity found in newspaper articles and the comments on them, revealing a higher degree of diversity in the latter.

[1] theguardian.com.

[2] We believe this is a valid methodology for topic modelling as it has been shown that prompting LLMs can obtain comparable results to more traditional methods such as Latent Dirichlet Allocation (Mu et al., 2024).

The particular selection of units described here (comments and paragraphs) is somewhat arbitrary; in Appendix A.1 we show that we get similar results by selecting other arbitrary units (sentences or clauses)

We calculate the topical diversity for each article in the SENSEI dataset and the comments section which accompanies it. We acknowledge that the following is a crude measure of topical diversity, which has been studied in more depth elsewhere (e.g. Azarbonyad et al. 2017; Bu et al. 2021). However, we believe this measure suffices to make what we consider an uncontroversial and intuitively correct point: that online comments are more topically diverse than news articles. [3]

Figure 1.2 shows the results of calculating the topical diversity for all articles and comments sections within the SENSEI corpus. Overall, the comments sections had roughly 1.3 times as many unique topics when compared to the articles: a Welch's t-test revealed that this difference was significant, $p < 0.001$. The effect size (Cohen's d) is 1.83.

Example 1.1: "Off-topic" comment in a comment section about bin collection schedules

User A: Sadly we already have two new plagues let loose upon the earth, they are called overpopulation and overconsumption.

This difference is hardly surprising, since newspaper articles tend to be well edited texts which are written strictly with the aim of sharing a single given story or opinion, whereas a comments section is a text written collaboratively by a number of different authors, who may enter the interaction with different interests and goals.

Qualitatively anaylsing the comments, we can see that some of the diversity may come from off-topic posts. Example 1.1 shows a comment which appeared below a newspaper article discussing bin collection policies, and is

[3] Our full methodology for acquiring the topics to calculate this index is detailed in Appendix A.1.

almost entirely irrelevant to the preceding discussion [4]. It appears that this user simply posted a comment on a topic that was more of interest from their point of view than the one under discussion previously.

1.3.2 Nested Argumentation in Online Debate

Another way in which new topics can arise during a discussion or debate is through what we will refer to as "nested argumentation", i.e. debates about the premises of debates[5]. Consider the case where speakers a and b are arguing about proposition p. Speaker a explains that she supports p because q. However, speaker b disagrees with q, and now the discussion has moved on to arguing about q instead of the original topic p. This process can repeat recursively until the dialogue has moved on to discussing premises which are far removed from the original topic. It is likely that this phenomenon is more prevalent in online dialogue than in face-to-face conversation, since face-to-face dialogues are generally limited to a small number of participants and have time constraints, whereas online dialogues are asynchronous and can have a much larger number of participants (potentially in the thousands for more popular websites).

Example 1.2 shows this kind of nested argument occurring in a real online dialogue. User A puts forth a comment arguing about the proposition "Britain needs a new aircraft carrier". One of the premises of their argument is "Britain does not need to project military power overseas". User B's subsequent com-

[4] Arguably this comment could considered "on topic" if we propose an enthymeme such as "so with more people consuming more and producing more waste, we need more frequent bin collections not less". However, in context we read it as an aside, and the replies to this comment did not return to the main discussion topic.

[5] This is a sub-type of what is commonly referred to in the Argument Mining literature as "serial argumentation" (see Freeman 1991, p. 2). However, "serial argumentation" refers to any chain of three or more propositions linked in series, regardless of whether they attack or support each other, whereas here we specifically address chains consisting of arguments that attack another argument's premises.

Example 1.2: A Nested Argument

User A: Precisely. This is an unnecessary behemoth ordered at the height of the Iraq lunacy that the coalition should have cancelled. The only point of large aircraft carriers is to project military power overseas. This we do not need to do.

User B: If we don't need to project military power overseas...then we don't need any armed forces at all do we? They're hardly likely to be deployed at home! Obviously if you consider it likely that the next 50 years won't require any Protection of vital shipping/trade routes.Support of Allies overseas. Maintaining treaty obligations. Commitment of forces as part of a multinational force. Defence of vital interests/sovereign territory. Urgent interventions against aggressor states. Humanitarian Interventions Etc etc...then we wouldn't need any armed forces...but its not very likely in an increasingly chaotic and unstable world is it ;)

User C: Unfortunately, our military adventures in the past decade have greatly contributed to the instability.

ment argues against this premise, and User C's reply to User B argues for it. In this way, the argument has moved on to debating User A's premise rather than the original topic.

1.3.3 Repetition

Online comments sections may also contain a high degree of repetition in the arguments which are presented — more so than texts produced by a single author — due to the fact that there are multiple participants in the discourse, not all of whom will carefully read all preceding comments to ensure that their contributions are unique. Therefore, a desirable feature of an argument summarisation system is that it can accurately group together comments which are making similar or identical points prior to summarising them. Example 1.3 shows two online comments with a high degree of overlap, which were

Example 1.3: Two comments with significant semantic overlap. Sections with similar meaning are highlighted in the same colour.

User A: Ever heard of an incubation period? "Border control" is a clearly a knee-jerk not based on an informed view of how to control a virus. Given the poster also says the UK's ONLY priority should be avoiding importation, it's pretty safe to say their agenda is more anti-foreigner than anti-virus (a) because control of the virus abroad is in our interests too and (b) because it's perfectly possible to focus on preventing spread to the UK and care about the impact elsewhere - they're not mutually exclusive.

User B: Border controls only help to pick up people who are symptomatic. It is possible to pass on the infection during the incubation stage. Therefore if an index patient travels to Paris and rides on a bus, before you know it you have infected weekend holiday makers returning home to the Common Travel Area. (CTA — I think I've just found a new name for the British Isles if Scotland gains independence — Ireland's already in the CTA). The best chance of fighting this is at source.

posted by separate users in response to the same comment on a thread about virus control.

1.3.4 Conclusions

In this section we have shown some of the complexities of summarising argumentative text. These problems all indicate that a different approach may need to be taken to this domain when compared to a monological text such as an essay or a newspaper article (where a conventional text-summarisation approach may be suitable). The fact that argumentative dialogues contain recursive (nested) argumentation, in particular, suggests the potential utility of a graph-based approach to dialogue summarisation. Such an approach would be able to capture the complex relationships between different argument components in a way that might be difficult to do succinctly using a purely text-

based summary.

1.4 Publications Arising from this Thesis

- Jonathan Clayton and Robert Gaizauskas. Predicting the Presence of Reasoning Markers in Argumentative Text. In *Proceedings of the 9th Workshop on Argument Mining*, pages 137–142, 2022
- Jonathan Clayton, Marco Damonte, and Robert Gaizauskas. Parsing Graphical Summaries from Argumentative Dialogues. In *Computational Models of Argument*, pages 37–48. IOS Press, 2024
- Jonathan Clayton, Marco Damonte, and Robert Gaizauskas. SENSEI-ASG: A Challenging Dataset for Argument Summary Graph Parsing. In *Proceedings of the 15th Language Resources and Evaluation Conference (LREC 2026)*, 2026. Accepted for publication

1.5 Chapter Overviews

The following is an overview of each of the chapters contained in this thesis. We note any chapters where a significant amount of work has appeared in a previous publication (all publications listed in Section 1.4).

- Chapter 2, **Background: Argumentation and Natural Language Processing**

  We review the important background areas for this thesis, including several subfields of Argument Mining and Natural Language Processing.

- Chapter 3, **Background: Argument Structure Parsing**

  We provide a more in-depth review of Argument Structure Parsing (ASP), an AM subfield which is fundamental for understanding this work.

- Chapter 4, **Reasoning Marker Prediction**

  In this chapter, we propose the task of Reasoning Marker Prediction. We view this task as a part of a possible pipeline for generating textual summaries from an Argument Summary Graph. We create a dataset for the task and investigate some Large Language Model-based approaches. *Parts of this chapter have been published in **Clayton and Gaizauskas (2022).***

- Chapter 5, **Argument Summary Graph Parsing**

  In this chapter, we introduce our novel task, Argument Summary Graph Parsing (ASGP) which we investigate in more detail in subsequent thesis chapters. As well as giving a basic outline of the task and our motivations for introducing it, we describe the metrics which we use to evaluate the proposed task. *Parts of this chapter have been published in **Clayton et al. (2024).***

- Chapter 6, **Preliminary Experiments on ASGP in Online Debate**

  This chapter describes the creation of a dataset for the task of ASGP proposed in the previous chapter, which we call Debatabase-ASG. We additionally carry out experiments with two different approaches to ASGP (both involving Large Language Models). The first approach is a two stage pipeline consisting of summarisation and stance detection components, and the second is using an end-to-end text-to-text model. We find that the end-to-end model is the better performing one for this task. *Parts of this chapter have been published in **Clayton et al. (2024).***

- Chapter 7, **End-to-end Techniques for Argument Structure Parsing**

  In this chapter, we use end-to-end models, similar to those used in the previous chapter, for the task of Argument Structure Parsing. We investigate the use of two different LLMs and two different output formats for this task. We find that our end-to-end models perform close to the state of the art for this task. Additionally, we find that the novel output

format we use has comparable performance to that used in previous work (Kawarada et al., 2024), while being significantly more succinct and therefore faster to process with an LLM.

- Chapter 8, **SENSEI-ASG: A Dataset for Argument Summary Graph Parsing of Online Comments**

  This chapter describes the creation of a second dataset for Argument Summary Graph Parsing, which we call SENSEI-ASG. This is corpus is derived from the SENSEI annotated corpus (Barker et al., 2016) which contains online comments from the Guardian news website.[6] *Parts of this chapter have been accepted for publication in **Clayton et al. (2026).***

- Chapter 9, **A Multi-Corpus Evaluation of ASGP**

  In this chapter, we carry out experiments evaluating the transferability of models trained on the two different dialogical datasets which we created in earlier chapters (Debatabase-ASG and SENSEI-ASG) as well as models trained on a monological Argument Structure Parsing corpus, the Argument Annotated Essays Corpus Stab and Gurevych (2014a), for the Argument Summary Graph Parsing task. We find that while training on additional in-domain data improves performance most on SENSEI-ASG (8.42% average improvement across the three metrics we used), training on the out-of-domain data also provided a reasonably large performance benefit of 8.11% on average. *Parts of this chapter have been accepted for publication in **Clayton et al. (2026).***

[6] theguardian.com

Chapter 2

Background: Argumentation and Natural Language Processing

2.1 Introduction

This thesis contains two background chapters: this chapter, which provides a general context in which the work is situated, and Chapter 3, which provides detailed background on a specific subtask of AM which is of particular importance to this thesis, Argument Structure Parsing or ASP.

In this chapter, we take a broad view of all subfields of Argument Mining which are relevant to this work. Firstly, Section 2.2, **Overview of Argument Mining**, summarises AM as a whole. Next, in Section 2.3, **Argument Mapping**, we look at the ways in which arguments have been mapped/diagrammed in past AM work, which is of importance for the choice of graphical scheme we use in this thesis.

Section 2.4, **Argument Summarisation** surveys relevant work that tries to summarise argumentative text, either in a graphical or textual format. In Section 2.5, **Dialogic Argument Mining**, we review work relevant to ours that looks at argument in dialogic text as we do (specifically those which are neither related to summarisation nor works which fall within the subfield of

ASP, reviewed in the next chapter).

In Section 2.6, **NLP Models and Techniques**, we give some background on specific techniques and models from the fields of Natural Language Processing and Machine Learning which are pertinent to the work carried out in this thesis. Finally, in Section 2.7, **Future Directions in Argument Mining**, we address recent directions in AM that may have a bearing on future work on the topic of this thesis, but which we have not directly addressed in our own research.

2.2 Overview of Argument Mining

The topic of this thesis, Argument Summarisation, depends on work carried out within the field of Argument Mining (AM). In their review of the field Lawrence and Reed (2020, p. 765) define AM as "the automatic identification and extraction of the structure of inference and reasoning expressed as arguments presented in natural language". The authors provide an example of a four-stage pipeline that could be used to carry out the complete task of AM:

(a) Text Segmentation

(b) Argument/ non-argument classification

(c) [Extracting] simple structure

(d) [Extracting] refined structure

In other words, the text is (a) first split into (potential) argumentative components (see Section 2.3) before (b) these are classified as argumentative or non-argumentative, and (c) argumentative links between them, such as "support" or "attack" are identified. In a final step, (d) (which we do not focus on

in this thesis) arguments can be analysed in further detail using, the theory of Argument Schemes (Walton, 2008), as in Visser et al. (2021).

These goals have defined much of the work in the AM field. Tasks (a) – (c) are often collectively referred to as Argument Structure Parsing (ASP). ASP will be discussed in more detail in Chapter 3.

However, there is also a significant amount of work in AM which does not fit neatly into ASP: in the words of Stede et al. (2019, p. 57), AM is a "constellation of subtasks". There is also significant amount of adjacent work analysing argumentative text which does not fit into this definition, since it either processes argumentative text without specifically analysing its structure (e.g. Argument Search; Stab et al. 2018b) or it makes use of argument structure for some goal separate from the identification of argumentative structure for its own sake (e.g. persuasiveness evaluation, Habernal and Gurevych 2016). In our discussion, we will also include work like this under the umbrella term of Argument Mining, due to the fact that much of this work uses similar techniques to AM proper, and has emerged from the same research community.

2.3 Argument Mapping

The idea of Argument Mapping, or (manually) creating a graphical structure that summarises the relations between the different components of an argument, is fundamental to the whole field of Argument Mining, since it underlies the creation of any AM dataset. For a fairly complete history of this field, see Stede et al. (2019, Chap. 3). Below we will review some of the more commonly used Argument Mapping schemes, before looking at Argument Mapping in dialogue, and finally at the notion of an Argumentative Discourse Unit, an important notion underlying Argument Mapping.

2.3.1 Argument Mapping Schemes

One of the most influential schemes for diagramming arguments comes from the philosopher Stephen Toulmin, in his book *The Uses of Argument* (Toulmin, 2003). In this book, he outlines multiple argumentative *components*, and describes the relations which can obtain between them.

Importantly, Toulmin's scheme, referred to in later work as the "Toulmin model of argument" is *informal*; it differs from syllogistic logic in that we do not have a set of premises that must deductively lead to a conclusion, but rather a model of the arguments used in everyday conversation and reasoning, which are often inductive rather than deductive and may be based on a large number of common-sense assumptions.

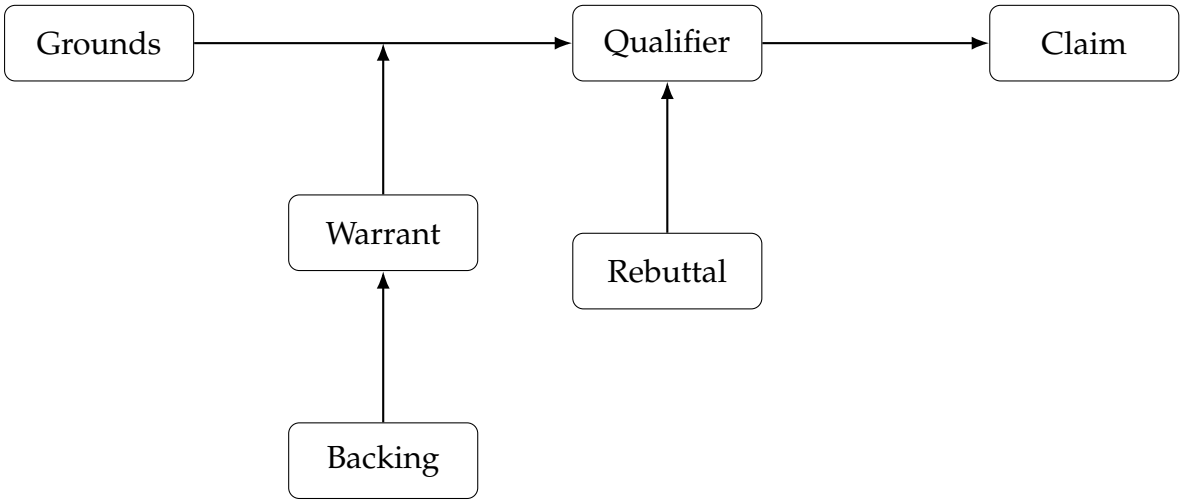


Figure 2.1: Toulmin Model Diagram

Another important element of Toulmin's work is his identification of the six main components which he argues make up informal arguments: claim, grounds, warrant, qualifier, rebuttal, and backing. While some of these elements are uncontroversial ("claims" are generally present in all argumentation theories, and likewise "grounds" are a particular subtype of "premises",

which are also uncontroversial), the other components are more controversial, particularly the “warrant”, which is the most important innovation of Toulmin’s theory. Warrants are “inference rules”, which are typically part of a person’s tacit knowledge, which allow people to infer a “claim” on the basis of the “grounds”. The notion is similar to the idea of an “enthymeme” or an unstated premise, in classical rhetoric (Aristotle, 1991), although the warrant may sometimes be stated explicitly. (One example that Toulmin uses is the warrant “A man born in Bermuda will be a British subject”, which can be used to draw the conclusion “Harry is a British subject” from the grounds “Harry was born in Bermuda”.)

Freeman (1991, 2011) objected to including warrants in argument maps, on the basis that they are mostly left unstated within actual argumentative texts. Freeman argued that this aspect of Toulmin’s theory reflects the fact that it is a theory of argumentation as a *process* rather than a *product*. In other words, Toulmin’s model is a tool for thinking through and analysing fully the structure of a given argument. This is useful for philosophers and other researchers who are composing arguments, but it does not necessarily reflect the elements which will typically be present in a spoken or written argumentative text.

Freeman does, however, incorporate a number of features from the Toulmin model into his argument structure formalism. For example, Qualifiers are introduced into the argument scheme as optional links between premise and conclusion. Freeman also incorporated and elaborated on Toulmin’s idea of the “rebuttal", distinguishing between two types of counter-arguments. Like Toulmin, Freeman’s model incorporates a type of counter-argument which attacks the inferential link between the premise and conclusion. Toulmin named this type of attack a “rebuttal" but Freeman renamed it to an “undercutting" attack. The term “rebuttal" is repurposed to refer to attacks which attempt to directly refute either a conclusion or a premise. Freeman’s distinction between these two types of attack is now standard within the argumentation commu-

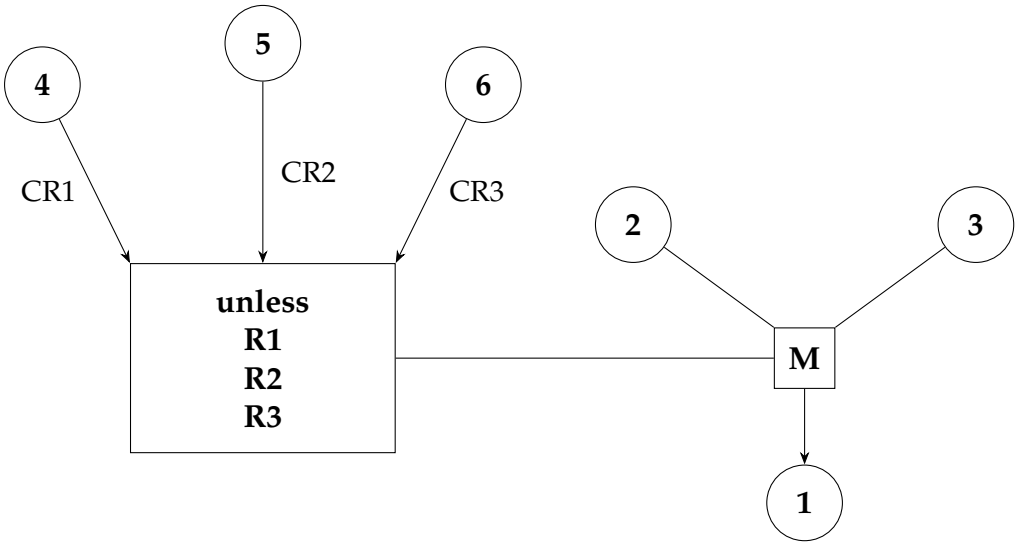


Figure 2.2: An example of an argument graph using Freeman's (1991) formalism of argument structure, illustrating the types of argumentative components and relations available in his formalism. In this argument, the premises ② and ③, are used to support a conclusion ①. M is a modal (similar to a "qualifier" in Toulmin's account), such as "possibly" or "probably". R1... R3 are rebuttals to the argument on the right, and ④, ⑤ and ⑥ are "counterrebuttals" responding to each of these rebuttals respectively (hence the labels CR1... CR3)

nity (see e.g. Rocci and Lucchini 2025). The Argumentative Microtext corpus (Peldszus and Stede, 2015a) is annotated using Freeman's annotation scheme, and unlike most other corpora, contains undercutting attacks.

2.3.2 Argument Mapping in Dialogue

One relatively recent innovation in Argument Mapping is Inference Anchoring Theory or IAT (Budzynska et al., 2016). Unlike other theories, IAT applies specifically to dialogue, and, as its name suggests, is a theory of how inferences are generated within dialogue as well as a graphing formalism. IAT differs from other formalisms in that it models both the argument structure of propositions within a dialogue and the "illocutionary forces" of statements. Illocutionary forces are a notion from the Speech Act theory developed by Austin and Searle (Austin, 1962; Searle, 1969) and are defined as a specific

kind of "act" performed by utterances in a dialogue, such as "asserting" or "questioning". This mapping of illocutionary forces to argument components suggests that the latter could at least partially be predicted from the former by an automatic system. For example, if Speaker A makes an utterance *asserting* the proposition p, which Speaker B then *questions*, it might be possible to infer that if Speaker A makes a subsequent utterance *asserting* q, then q is a premise which supports p (see Budzynska et al. 2016, p. 93, for a further explanation).

Figure 2.3 illustrates this type of dialogue in the IAT formalism. IAT analyses dialogues using three different "layers". From right to left, these are the the discourse layer, the illocutionary layer and the argumentation layer.

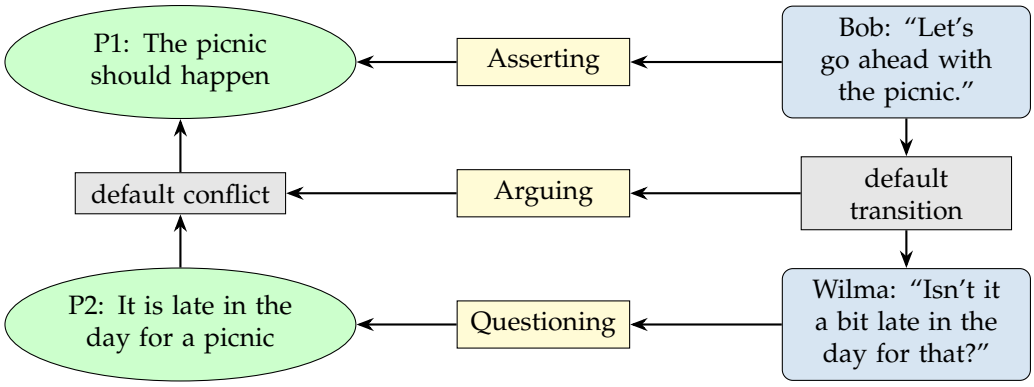


Figure 2.3: An example IAT diagram

The discourse layer contains the actual text of the dialogue, segmented into locutions, which consist of a single ADU and the name of the speaker. The ADUs are segmented such that they contain a single argumentative proposition. In addition to locutions, there is a "transition" node between each sequential pair of locutions. This is a proposed latent entity which is important in IAT theory, because it is proposed that inferences can arise not just from utterances, but from the transitions between them.

The leftmost layer, the argumentation layer, is an argument graph similar to those we have examined previously in Section 2.3, consisting of proposi-

tions and relations between them, which have types including "attack" and "support" (and sometimes other relations such as "rephrase").

2.3.3 Argumentative Discourse Units

One of the most important notions in Argument Mapping is that of the Argumentative Discourse Unit or ADU. The term was coined in Peldszus and Stede (2013), in reference to the concept of an "EDU" or "Elementary Discourse Unit", which is the fundamental unit of text within discourse analysis that was proposed by Rhetorical Structure Theory (Mann and Thompson, 1988). In analogy to EDUs, ADUs are the basic units from which an argumentative text is composed. Quite often, these units are defined to have a "type" such as "premise" or "conclusion".

However, there are several differences between the two: firstly, while EDUs are typically a single clause, ADUs can be of any length and do not depend strictly on syntactic features: a multi-sentence premise or conclusion can be an ADU (Peldszus and Stede, 2013). Additionally, while every token of a text that has been annotated using Rhetorical Structure Theory will belong to an EDU, this is not the case with ADUs, since not all texts are purely argumentative, and argumentative texts may contain non-argumentative spans.

As a consequence, annotating ADUs within text is a difficult task: annotators must not only agree on the type of ADU component and its relation to others, but also where the boundaries of the components are, and unlike with EDUs, they lack even the guidance of syntax and punctuation which show where clauses start and end. In practice, in order to sidestep this difficult and expensive issue, many authors simply use either sentence boundaries (e.g. Habernal and Gurevych 2017) or clause boundaries (e.g. Palau and Moens 2009) as the unit of segmentation.

Corpus Type	Corpus Name	Size (words)
Argument Structure Parsing	Argument Annotated Essays (AAEC) (Stab and Gurevych, 2014a)	147,271
	Microtext Corpus (MTC) (Stede et al., 2016)	7,828
	Consumer Debt Collection Practices (CDCP) (Park and Cardie, 2018)	~88,000
	European Court of Human Rights (ECHR) (Poudyal et al., 2020)	~300,000
	Argument Annotatred SciDTB (Accuosto and Saggion, 2019)	~8,000
	ICNALE-AS (Putra et al., 2022)	~100,000
Dialogic AM	ComArg corpus (Boltužić and Šnajder, 2014)	~265,808
	US2016 (Visser et al., 2019)	17,190
	Internet Argument Corpus (IAC) (Abbott et al., 2016)	~73,000,000
	Question Time Corpus (QT30) (Hautli-Janisz et al., 2022)	~280,000
	CMV Modes Dataset (Chakrabarty et al., 2020)	~330,000
Summarisation	SENSEI (Barker et al., 2016)	87,559
	DebateSum (Roush and Balaji, 2020)	~101,000,000
	VivesDebate (Ruiz-Dolz et al., 2021b)	139,756
	Argument Annotated User-Generated Web Discourse (Habernal and Gurevych, 2017)	84,673
	Key Point Analysis Dataset (Bar-Haim et al., 2020)	445,012

Table 2.1: Overview of the corpora mentioned in this chapter and Chapter 3

2.3.4 Argument Mining Datasets

Table 2.1 contains a summary of the Datasets/ Corpora that we discuss in this and the following chapter. We have divided the table into three sections: Summarisation of Argument (Section 2.4), Dialogic AM (Section 2.5), and Argument Structure Parsing (Chapter 3).

2.4 Argument Summarisation

The works that we will refer to here as "Argument Summarisation" form a loose collection of work rather than a coherent field, since researchers do not necessarily agree on how to go about summarising an argument. Not all of the tasks described here are necessarily referred to as "summarisation" by the authors of the original research. We try to collect all relevant research which aims to do either of the following:

(a) create a textual summary of an argument (i.e. a summary consisting of free text only, without any structured data in the output)

(b) compress a lengthy argument by some other means, e.g. clustering related points, or building a simplified graphical summary

We review these two different approaches separately in the two following subsections.

2.4.1 Textual summarisation

Egan et al. (2016) is an early paper addressing the task of Argument Summarisation. It compared multiple approaches to summarizing argumentative dialgoues: a simple sentence extraction based approach (Nenkova et al., 2006), and their own "point-based" approach. They define "points" as recurring propositions that appear within a debate, which they detect using a combination of dependency parses and verb frames (Baker et al., 1998). Their pipeline first extracts the points from the debate, before clustering them based on similar semantics, and then finally creates a human-readable summary of the debate as a whole by placing the collected point clusters into multiple categories, e.g. "People disagree on these points:", and "Users that talk about point X often talk about point Y".

Barker et al. (2016) create a dataset called SENSEI, based on comments on the *Guardian* news website, for the tasks of argument mining and summarisation. The dataset has several types of annotations which are useful for summarising argument, including text-based abstractive summaries of the entire comment thread, as well as clusters of comments with corresponding topic labels, and per-comment summaries.

Roush and Balaji (2020) generate a corpus called DebateSum, and another named OpenDebateEvidence (Roush et al., 2024). While these are both useful resources for textual summarisation, the corpus is not geared towards summarising debates themselves, but rather towards extracting summaries from "evidence" documents which could be deployed in a competitive debate.

Ruiz-Dolz et al. (2021b) create a corpus of competitive debate in Catalan, which the authors claim can be employed for a number of purposes including summarisation. The summaries are overall summaries of a "side" in a debate, created by the participants themselves in the "summing-up" phase which oc-

curs towards the end of a live debate.

Irani et al. (2024) create a task/framework for summarising short argumentative texts such as comments. They use a pipeline of models consisting of argument detection followed by Claim Topic Extraction and finally Argument Stance Classification. In this way they generate a formulaic, structured summary of a short argumentative text along the lines of "the text is an argument about topic T with stance S". This somewhat differs from the type of summary which we are aiming to produce, since we focus on informative, rather than indicative summaries (Hahn and Mani, 2000).

Out of the works reviewed in this subsection, only the SENSEI dataset Barker et al. (2016) is directly relevant for our purposes, since, as we will explain in Chapter 5, we are interested in generating a representation of argumentative dialogues consisting of a graph that contains textual summaries of individual comments. In Chapter 8, we will describe how we adapt the data in SENSEI to create our own dataset for this purpose.

2.4.2 Graphical Summarisation

One early work looking at graphical argument summarisation is Misra et al. (2017). This work uses human text summarisation as part of a pipeline aimed at discovering argument "facets", or, in other words, typical sub-topics that people will often discuss when debating a given main topic. For example, people debating the topic of "nuclear energy" will often discuss the facet of "climate change". In order to discover these facets, text summarisation is first used to extract a single "central proposition" per comment in an online debate, and then a clustering algorithm is used to group these propositions into facets based on their semantic similarity. This differs from our work in that they do not attempt to detect argumentative relations between the propositions.

Habernal and Gurevych (2017) annotate 340 short web documents (for example online comments and blogs) using a modified version of the Toulmin model. They do not annotate relations between ADUs; however, they only allow one "claim" per document, and therefore all ADUs are implicitly linked together due to the constraints present in the Toulmin model (for example, the "warrant" must link the "grounds" to the "claim"; see the explanation and diagram in Section 2.3. Their work has a different focus from ours since they look at single comments in isolation instead of in the context of a dialogue.

Bar-Haim et al. (2020) create a task called "key point analysis", a form of argument summarisation. In their formulation of the summarisation task, instead of summarising comments directly, they attempt to design a system that maps comments to what they call "key points", or brief summaries of arguments which tend to recur when discussing a given topic. Their approach therefore presupposes that the argument topic has already been identified, and a list of key points for the topic exists, to be able to generate a summary for a given argument. Despite this limitation, the dataset that they produce is a useful resource.

Diaz et al. (2022) use data from Twitter to generate what they call "stance trees", which are trees containing clusters of twitter comments that may have three polarities towards their parents in the stance tree, either positive, negative or uncertain. Their work differs from ours in that the trees they produce do not have a per-node summary, but otherwise the representation that their algorithm produces is similar to those that we will attempt to generate later in this thesis.

Barrow et al. (2021) study cross-document argument mining in document collections. This work has two main innovations: firstly, they define their own task of "viewpoint reconstruction", which involves extracting what they call a "viewpoint" from a text. These are 3-tuples consisting what they call a

“topic", an “aspect" and a “stance", in that order. For example, the viewpoint V = (Nuclear Energy, environmental impact, PRO) describes a claim made in a document in favour of nuclear energy, addressing the “aspect" of environmental impact. Secondly, they create the concept of a “syntopical graph”, a type of data structure engineered for this task, which encodes the relationship between different documents and the viewpoints contained within them. It is a multigraph with two types of nodes: documents and claims. There are different edge types: firstly, edges from the documents to claims contained within them, and secondly edges between the claims which show semantic relations between them such as entailment or support. They use this data structure as an input to a relational Graph Convolutional Network (Schlichtkrull et al., 2018) in order to predict viewpoints.

Altemeyer et al. (2025) carry out a systematic study on the use of LLMs in the task of argument summarisation. They are inspired by the work of Bar-Haim et al. (2020) (summarised in above), but give a more general formulation of the argument summary task. Specifically, given a set of k arguments on topic t with a given perspective (typically, pro or con), the task is to produce a set of N summaries (and optionally, ratings of argument strength), where $N \ll k$. They additionally formulate a taxonomy of argument summarisation systems, dividing them into *classification-based* systems and *clustering-based* systems. *Classification-based* systems, like the one devised by Bar-Haim et al. (2020), have a set of typical arguments for a given topic, which have been generated *a-priori*, whether through crowdsourcing or by domain experts. The system then works by matching individual argumentative texts (such as comments) to these typical arguments. *Clustering* systems, on the other hand, first group the comments using a generic clustering algorithm - for example, Khosravani et al. (2024) used the HDBSCAN algorithm (McInnes et al., 2017) - before using extractive or abstractive summarisation techniques to summarise each cluster.

From the papers reviewed in this section, the work that we carry out in this thesis is closest, in terms of its goals, to that of Diaz et al. (2022). Like this work, we are interested in analysing the internal structure of a dialogue in order to produce a useful summary. The stream of work represented by Bar-Haim et al. (2020) and Altemeyer et al. (2025), while potentially relevant in terms of the techniques they use, has the goal of exhaustively enumerating *all* possible arguments on a given topic; this is not the focus of this thesis.

2.5 Dialogic Argument Mining

In this section, we review work that has been done on Dialogic Argument Mining, aside from that which directly involves either summarisation or Argument Structure Parsing, as we review these two topics elsewhere. It is important to mention here because the annotation schemes used by many of these papers share similarities with those used in our work (see Chapter 5).

Boltužić and Šnajder (2014) is an early work looking at dialogic AM. In this study, the authors take online forum comments and annotate them with a set of pre-defined viewpoints, and set the task of detecting the correct stance label {strongly support, support, neutral, attack, strongly attack} for each comment-viewpoint pair.

The Internet Argument Corpus Abbott et al. (2016) is an early corpus for the task of dialogical argument mining. Data from online discussion forums is used. The most relevant annotations for our perspective are the support/attack relations.

Chakrabarty et al. (2020) create a dataset based on the changemyview subcommunity on Reddit [7], an online debate forum. Named "Change my view

[7] reddit.com/r/changemyview

modes", it has a more fine-grained annotation structure than the IAC. Both inter- and intra-comment relations are annotated, but inter-comment relations are annotated between specific ADUs within comments rather than between comments as a whole, reflecting the fact that, in an online setting, it is possible for a lengthy comment to address multiple parts of the comment that it responds to, possibly agreeing with one point and disagreeing with another.

Much work has been done, beginning with Budzynska et al. (2014a), to annotate texts according to Inference Anchoring Theory (an annotation scheme combining illocutionary relations with argumentative relations, described in Section 2.3). The QT30 corpus (Hautli-Janisz et al., 2022) is the largest corpus annotated using Inference Anchoring Theory. The corpus is derived from discussions which took place on the British political debate television program *Question Time*.

Ye and Teufel (2024) collect a dataset from `kialo.com` which they use for dialogic Argument Structure Parsing. The novel aspect of their dataset is that instead of just "support" and "attack" relations, they also annotate, and train a model to detect, "undercuts", or attacks to inferences between different argument components (see Section 2.3).

The annotation scheme that we use in our thesis (which we introduce in Chapter 5) is a simple annotation scheme consisting of support and attack relations between comments, similar to those used by Boltužić and Šnajder (2014) or Abbott et al. (2016). However, the datasets that we produce in this thesis could in principle be extended to either include ADU-level annotations like those in Chakrabarty et al. (2020), or to add extra relation types as Ye and Teufel (2024) do, and these research directions are worth pursuing in future work.

2.6 NLP Models and Techniques

In this section, we briefly review some NLP models and techniques that will be used in this thesis. Firstly, in Section 2.6.1, we briefly review Large Language Models (LLMs) and their use in Argument Mining. In Section 2.6.3, we describe a Parameter Efficient Fine Tuning (PEFT) method that we use in some of the following chapters. Finally, in Section 2.6.4, we review work that investigates the limitations of LLMs within Argument Mining.

2.6.1 Large Language Models

Large Language Models are neural network models applied to language-related tasks, which have a large number of learnable parameters (usually in the millions or more), and have typically been trained on a large quantity of text data. While neural networks have been applied to language tasks since the 1980s (Jurafsky and Martin, 2023, Chapter 7) the volume of research and the performance of these models across a wide variety of tasks has increased dramatically in the last decade. Key to this success has been the ever-decreasing cost and performance of Graphical Processing Units which are used to train the models, enabling model sizes to dramatically increase, as well as a number of technical innovations, notably the development of Transformer models (Vaswani et al., 2017). A good summary of the various architectures available, training methods, inference, and other details of these models, is found in Jurafsky and Martin (2023, Chapter 10).

Pretrained Large Language Models are now state-of-the-art in a number of the subfields of Argument Mining described earlier in the chapter. Some examples include argument component type classification (Al Zubaer et al., 2023), relation type classification (Gorur et al., 2024) and argument generation (Chen

Model and Reference	N. Trainable Parameters	Architecture
BERT (Devlin et al., 2018)	110M, 340M	Encoder-only
Longformer (Beltagy et al., 2020)	149M	Encoder-only
Flan-T5-XXL (Chung et al., 2022)	11B	Encoder-Decoder
ChatGPT-3.5 (OpenAI, 2022)	Approx. 175B	Decoder-only
LLaMA-2 (Touvron et al., 2023b)	7B to 70B	Decoder-only
Phi-2 (Bai et al., 2023)	1.1B	Decoder-only
LLaMA-3 (AI, 2024)	8B, 70B, 405B	Decoder-only

Table 2.2: LLMs used in this thesis

et al., 2023). LLMs have also been used extensively in Argument Structure Parsing, as we will cover in the following chapter. A list of the LLMs which we have used in this thesis is shown in Table 2.2.

2.6.2 In-Context-Learning

In recent years, several "learning paradigms" have emerged for Large Language Models (Liu et al., 2021). This concept has emerged in the context of Large Language Models which have been pretrained on enormous corpora. The "pretraining" stage typically mostly consists of generic language-based tasks such as predicting the next token in a sentence. This yields LLMs which are good at modelling the structure of language, but will not necessarily perform well at the specific task that the user has in mind, the "target task". The question naturally arises of how to adapt these pretrained LLMs to produce a model which will perform well in the target task.

The most common of these, and one which we use in several chapters of the thesis, is the fine-tuning paradigm, in which we simply carry out further

supervised training of the model using data from the target task (Peters et al., 2019). However, there are several other possibilities for adapting a pretrained LLM. One of these, which we will investigate in Chapter 6, is In-Context-Learning or ICL (Brown et al., 2020).

In ICL, no fine-tuning is carried out, but rather the task to be performed is "demonstrated" to the model within the text prompt at inference time. For example, if the target task were "translating words from Quechua to English", an ICL prompt would contain several examples of inputs (Quechua words) and outputs (English words) prepended to the target word to be translated. Example 2.1 shows how such a prompt might be structured.

Example 2.1: Example of the structure of a prompt used in In-Context-Learning

Translate each of the following Quechua words into English. Give only the English translation.

Word: wasi
Translation: house

Word: yachachiq
Translation: teacher

Word: mikhuna
Translation: food

Word: wayna
Translation:

ICL has the advantage of being much less computationally expensive than fine-tuning, since we do not have to update the model weights at all. However, it seems to only be applicable to models above a certain size, and may also perform poorly on truly novel tasks where it is hard for a model to generalise to a new task using what it has "learned" from the pre-training data (Mosbach

et al., 2023).

2.6.3 Parameter Efficient Fine Tuning techniques for LLMs

Due to the large size, in terms of number of parameters, of modern LLMs, fine-tuning can often be expensive, both in terms of hardware requirements and training time. In this thesis, most work was carried out on High-Performance computing nodes, in which the maximum hardware specifications available were eight NVIDIA V100 GPUs, each with 32GB of VRAM.

Due to these hardware limitations, for training some of the larger models, it was necessary to take adavantage of what are known as Parameter Efficient Fine Tuning techniques or PEFT. A specific technique which we use in several chapters is QLoRA (Dettmers et al., 2023). QLoRA is a type of low-rank adaptation (Hu et al., 2021), a family of techniques which consist in adding a number of smaller "adapter" modules to the LLM which are updated during the backpropagation phase of training, while the original model parameters remain "frozen", i.e. they remain unchanged. In QLoRA, the base model is additionally quantized, i.e. model parameters are converted into a data type with only 4 bits, using an efficient quantization technique developed by the paper authors.

2.6.4 Limitations of LLMs in Argument Mining

Despite the promising performance of LLMs across a number of AM tasks, some authors have shown them to have limitations. An investigation into the reasoning of LLMs on Argument Mining tasks was carried out by de Wynter and Yuan (2024), who argue that LLMs are incapable of abstract reasoning due to the fact that small changes in the task setup (such as adding or removing

line numbers) can have a large effect on their performance.

Jakobsen et al. (2021) investigate several argument mining tasks, and find that LLMs rely on spurious correlations when reasoning about argument. They use the LIME model interpretability tool (Ribeiro et al., 2016), which generates "local explanations" of a specific decision made by a classifier, via fitting a logistic regression model to other model outputs in the neighbourhood of the decision, to train a DNN. They investigate several Argument Mining tasks including Argument Pair Extraction (Cheng et al., 2020) and sentence classification. Using LIME, they find that content words seem to be more important than function words when making a decision. For example, for the topic of gun control, they find that the inclusion of the word "gun" influenced the decision of whether to classify a sentence as for or against gun control much more than function words such as "therefore" or "because", which are largely ignored. They argue that, since it is hardly possible to understand an argument without considering function words, LLMs must be largely relying on spurious correlations when making decisions. However, they find that the bias in favour of content words is somewhat mitigated when using a MTL (Multi-Task Learning) setup, implying that MTL may have a regularising effect that helps LLMs avoid spurious correlations, which concurs with previous work by Tu et al. (2020) on MTL in a separate domain of NLP.

Niven and Kao (2019) look at the Argument Reasoning Comprehension Task (ARCT) (Habernal et al., 2018) a multiple choice task, in which the system has to choose between two different warrants (see the description of the Toulmin model in Section 2.3). Niven & Kao find that the strong performance of BERT-based models on this dataset was largely due to spurious statistical cues (notably, the presence of the word "not" in the correct warrant). They create an adversarial dataset in which these spurious cues are deliberately correlated with the correct option in the training set but the incorrect option in the test set, and they find that in this setting, BERT models perform no better than chance.

Mayer et al. (2020) show that NLP models used in AM can be vulnerable to "adversarial attacks", that is, small changes to the input to the models, such as an added punctuation mark, which would be ignored as a typo by a human reader, can change the label that is output by a model.

2.7 Future Directions in Argument Mining

Argument Mining is a growing field, with several promising research strands that fall outside the scope of this thesis and were therefore not addressed in the preceding sections. We nevertheless briefly outline these strands to situate our work within the broader research landscape and because of their potential relevance to future developments in graphical argument summarisation.

2.7.1 Encoding External Knowledge

One important direction of recent work has been attempting to encode external knowledge into Argument Mining (AM) systems, typically using either Knowledge Graphs (KGs) or other external data sources such as Wikipedia.

Knowledge Graphs Hogan et al. (2021) are a subtype of structured database that typically take the form of (h, r, t) triples, where h and t refer to two entities (the head and tail) and r to a relation or predicate that holds between them. One example of such a triple could be $(\text{London}, \text{CapitalOf}, \text{UK})$.

Domain-specific Argument Knowledge Graphs (AKGs) can be developed to represent entities often featured in debates. Al-Khatib et al. (2020) design one such type of AKG that aims to encode causality between common entities that appear in arguments. Two types of relation are included "positive" and

"negative", which refer to positive and negative causality. There are two types of entities, "concepts" and "consequences". A concept has a "positive" relation towards a consequence if it "promotes/ causes/ leads" to that consequence, and a "negative" one if it "suppresses/ prevents/ stops" that consequence. Nuclear power, for example, could have the following relations:

("Nuclear power", +, "Nuclear waste")

("Nuclear power", −, "Carbon emissions")

Applying Knowledge Graphs to AM There are multiple ways in which knowledge encoded in KGs can be incorporated into AM tasks. One common approach is through knowledge graph embeddings, learn continuous representations of entities and relations using scoring functions that capture structural regularities in triples. One simple model, TransE (Bordes et al., 2013), models relations as translations in embedding space:

$$\mathbf{h} + \mathbf{r} \approx \mathbf{t}. \tag{2.1}$$

This is operationalised by the scoring function

$$f(h, r, t) = ||\mathbf{h} + \mathbf{r} - \mathbf{t}||_p, \tag{2.2}$$

where $p \in \{1, 2\}$ denotes the L_1 or L_2 norm. Lower scores indicate more plausible triples.

Such KG embeddings can then be integrated into neural or LLM-based AM systems by combining textual representations with relevant KG-derived embeddings, as in Dore et al. (2025). Alternatively, relations extracted from a

KG can simply be encoded as text and used to augment the input to a language model, an approach adopted in Al Khatib et al. (2021).

Aside from Knowledge Graphs, there are a variety of novel ways in which researchers have attempted to combine external information sources with neural models when approaching argument mining tasks. One example of this is Gemechu and Reed (2019), a work tackling argumentative relation classification, in which text taken from Wikipedia that is relevant to the topic of an argument is fed into an LLM together with the text to be classified (we discuss this work further in Section 3.5.1).

2.7.2 Graph Neural Networks

Graph Neural Networks, or GNNs, are increasingly being applied to a range of Machine Learning tasks, including in NLP. These models are intuitively applicable to AM tasks where either the output is a graph or some information about graph structure is required to solve a task. While other types of NNs can represent or reason about graphs, GNNs are useful in that they explicitly constrain a model to use graphical representations.

Saveleva et al. (2021) use GNNs to encode global features about graph structure that can be used for the task of argument quality prediction. Similarly, Barrow et al. (2021) use argument graphs as an input structure to a GNN used for stance prediction.

GNNs can also be used without the structure being used as an input to the model; in Gemechu and Reed (2025) a GNN is used for the task of ASP. The rationale of the use of this model type is to constrain the network to learn a graph structure as an intermediate representation, even though the actual decoding of the graph structure happens in the model's classifier head.

2.8 Conclusion

In this chapter, we have reviewed several areas of Argument Mining relevant to our work, including Argument Mapping, Argument Summarisation, and Dialogic Argument Mining, as well as reviewing relevant NLP models and techniques, and finally looking at future directions in AM.

In general, this review has shown that there is a long tradition of research that aims to extract graphical argument structures from argumentative texts, both monological and dialogical. However, in all of these fields, the main focus has been the extraction of argumentation structures from text for its own sake, rather than for a specific goal. Our interest in developing applications which are useful for creating a readable summary of a debate has led to our proposing an alternative type of graphical representation, as we will explain in Chapter 5, **Argument Summary Graph Parsing**.

In Section 2.4, we have reviewed work on summarisation of argument, both textual and graphical, revealing that there is a growing body of literature on the topic. Increasingly, researchers are becoming interested in summaries which have some level of structure, with the simplest form of this being lists of arguments which are accompanied by a stance label ("pro" or "con") towards the topic in question. However, much of this work differs from the content of this thesis in two important ways. Firstly, it is often interested in extracting arguments from a large text collection, with the goal of examining every possible pro/ con argument on a given topic; this contrasts with our goal of summarising an individual dialogue to discover the views of a particular set of speakers. Secondly, these authors are typically not interested in extracting more fine-grained information from a given text (such as which arguments are used to attack or support others), since again they are most interested in exhaustively listing possible viewpoints on a given "issue", instead of analysing

a given text. Again, this motivates the novel task that we will propose and describe in detail in Chapter 5.

Chapter 3

Background: Argument Structure Parsing

In this chapter we will give an extended review of Argument Structure Parsing. We choose to focus on ASP at length because it is highly important background for the rest of this thesis. In particular, it is important for understanding the novel task of Argument Summary Graph Parsing (ASGP) which we develop and explore in Chapters 5, 6, 8 and 9, as well as Chapter 7, in which we investigate ASP directly.

This chapter has five content sections. Firstly, Section 3.1, **Monological Argument Structure Parsing**, we describe how the ASP task is usually defined for monological text. In Section 3.2, **Evaluation of ASP**, we describe how previous authors have evaluated the task.

We then review the corpora that have been created to advance research in ASP (Section 3.3, **ASP Corpora**), before moving on to the reviewing models/approaches used to tackle the task (Section 3.4, **Approaches to Argument Structure Parsing**), and finally, examining how ASP has been approached in the challenging domain of dialogical text (Section 3.5, **Argument Summary Parsing in Dialogue**).

3.1 Monological Argument Structure Parsing

Argument Structure Parsing (ASP) is the task of extracting a complete argument structure, of the type described in Section 2.3, from a text. To illustrate this task, we include representations of example inputs and outputs in Figure 3.1. This figure illustrates the overall structure of essays and corresponding argument structure trees in the Argument Annotated Essays Corpus (AAEC) created by Stab and Gurevych (2014a, henceforth the authors will be referred to as S&G for brevity). This is one of the earliest corpora developed for the ASP task, as well as one of the most thoroughly studied.

Example 3.1: An example of a body paragraph from the Argument Annotated Essays corpus, (text corresponds to Body Paragraph 1 in Figure 3.1)

Firstly, [by having CCTV cameras at workplace, crimes such as robbery can be trimmed down PREMISE]. This is because [when a shop is attacked by thieves, the clips taken by these cameras serve as a source of evidence to help the thieves to be traced PREMISE]. Not only that, [as the employees know that they are under constant surveillance, they will less likely steal from the shops PREMISE]. Thus, [it is clear that CCTVs must be put in all workplace so that crime rates can be minimized CLAIM].

As we can see in Figure 3.1, the AAEC annotation scheme contains multiple ADU types: "Major Claims", "Claims", and "Premises", which can be connected by four types of relation, including "Support", "Attack", "For" and "Against". This is only one of many possible ASP annotation schemes (as we will see below in Section 3.3); we explain it in detail here for illustrative purposes and because we carry out an investigation using this dataset in Chapter 7. S&G's scheme is specifically designed for argumentative essays, rather than other types of text, as can be inferred by their inclusion of a "Major Claim" ADU type, which it is reasonable to include when annotating a short essay

which tends to be focused around a single major claim, but not necessarily for other argumentative text types such as dialogues.

3.1.1 Subtasks of ASP

S&G divide ASP into four subtasks, which can be viewed as possible components of a pipeline approach to ASP. These are Argument Component Identification, Argument Component Type Classification (ACTC), Argument Relation Identification (ARI), and finally Argument Relation Type Classification (ARTC). These four components, while a common way to divide the task, are not used by every ASP system (with Argument Relation Identification and Argument Relation Type Classification, for example, often being combined into a single subtask, as in e.g. Kuribayashi et al. 2019).

Each of the final three steps has a corresponding evaluation metric; these metrics have been commonly used to evaluate ASP in the literature (see e.g. Gemechu and Reed 2025). These are Component-F1 (ACTC), Relation-F1 (RI) and Relation Type-F1 (ACTC). We describe these metrics in more detail in Section 3.2.

Argument Component Identification (ACI) is the task of splitting the text into "Argumentative Discourse Units" or ADUs. At this point, the ADU type (e.g. Claim or Premise) is not yet identified, but the non-argumentative text spans (which in the AAEC include text fragments such as "In conclusion", which lack propositional content), are discarded.

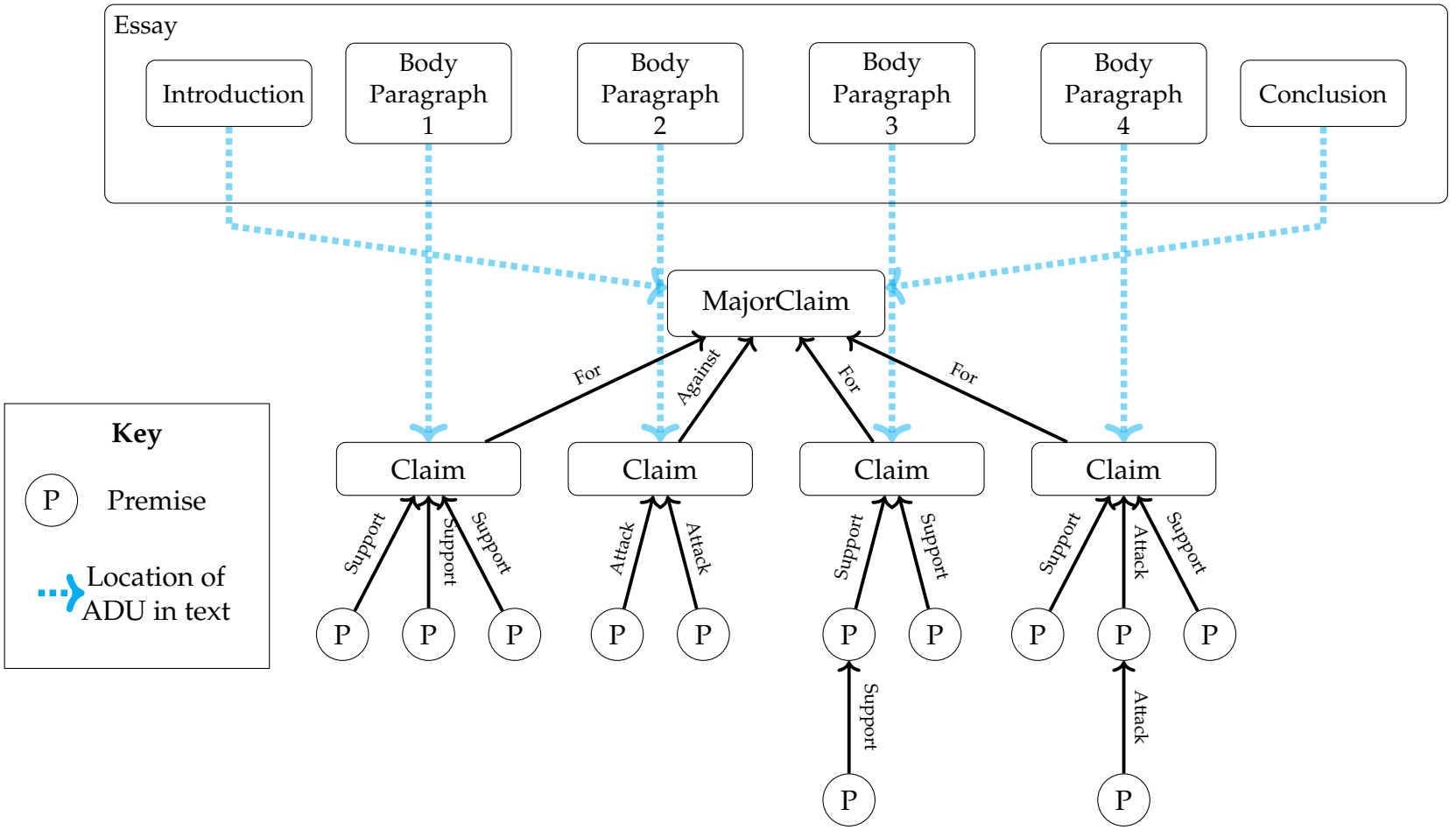


Figure 3.1: An example of the structure of an essay from the Argument Annotated Essays Corpus (Stab and Gurevych, 2014a) and a corresponding argument tree. The blue arrows show where the ADUs in the tree are located within the essay. Premises appear within the same paragraph as their corresponding claims.

Argument Component Type Classification (ACTC) is the task of classifying ADUs into a set of "types", which describe their function in the argumentative discourse. In S&G's scheme, these can be either "Claim", "Premise" or "Major Claim". Each essay has a single "Major Claim" and every paragraph in the essay has a single "Claim" which is supported or attacked by one or more "premises".

Argument Relation Identification (ARI) is the task of predicting (directed) links between the components identified in the previous step. Often restrictions exist between the types of components which may link to each other; in S&G's scheme for example, a Premise can only link to either another Premise or a Claim, and a Claim can only link to a MajorClaim.

Argument Relation Type Classification (ARTC) is the task of classifying the links identified by the ARI component into "types". In the case of S&G, there are two pairs of link types: "For" and "Against", and "Support" and "Attack", with the only difference between the two pairs being that the former apply to relations between Claims and Major Claims, while the latter apply between other pairs of ADUs.

3.2 Evaluation of ASP

In this section, we review how monological ASP has been evaluated. First, in Section 3.2.1, **Note on Terminology**, we describe how the terms we use for the metrics in this thesis differ from those used previously and why. Second, in Section 3.2.2, **Preprocessing Model Output for Evaluation**, we discuss how other researchers have tackled preprocessing model outputs prior to performing evaluation. Next, in Section 3.2.3, **Definitions of the Metrics**, we

formally define the metrics used for monological ASP. Finally, in Section 3.2.4, **Criticisms of ASP Metrics** we give some criticisms of these commonly-used metrics.

3.2.1 Note on Terminology

Within ASP, it has been common to use two different sets of metrics depending on the general class of model used — that is, one set of metrics for ADU-level models, and another for end-to-end models (we explain the difference between these two approaches in Section 3.4).

For the ADU-level models, the metrics used, originating in Stab and Gurevych (2017), have been simply referred to by their subtask name plus "macro-F1", i.e. ACTC Macro-F1, ARI Macro-F1, and ARTC Macro-F1. The metrics used for end-to-end models, originating in Eger et al. (2017), have been referred to as Component-F1 and Relation-F1 (this work lacks a specific metric for the ARI subtask, justifiably considering it redundant).

Despite the different naming conventions, however, these are in fact the micro and macro-averaged versions of the same metrics. Throughout this thesis, therefore, we adopt a simple and more concise naming convention, corresponding to the element type classified by each metric; for the macro-averages, Component-F1 (C-F1), Relation-F1 (R-F1), and Relation Type-F1 (T-F1), and for the micro-averages, the same names but with the word "Micro" added, e.g. Component-Micro-F1 (C-Micro-F1). Table 3.1 shows the metric names used in previous works, and the equivalents we use here.

Source	Subtask	Metric name	Equivalent in this thesis
Stab and Gurevych (2017) (ADU-Level)	ACTC	ACTC Macro-F1	Component-F1 (C-F1)
	ARI	ARI Macro-F1	Relation-F1 (R-F1)
	ARTC	ARTC Macro-F1	Relation Type-F1 (T-F1)
Eger et al. (2017) (End-to-end)	ACTC	Component-F1 (C-F1)	Component-Micro-F1 (C-Micro-F1)
	ARI	N/A	Relation-Micro-F1 (R-Micro-F1)
	ARTC	Relation-F1 (R-F1)	Relation Type-Micro-F1 (T-Micro-F1)

Table 3.1: Comparison of metrics used in prior work and their equivalents in this thesis.

3.2.2 Preprocessing Model Output for Evaluation

In this section, we will explain the necessity of preprocessing for calculating the metrics described in the following section. The definitions of the metrics that we will give below assume two argument graphs: a gold-standard graph, G_{true}, and a predicted graph, G_{pred}. The nodes in both graphs must consist of sets of unique labels, and furthermore the labels must be aligned correctly where the ADUs in the two graphs match.

Example 3.2: ADU Alginment Problem

Gold Standard ADUs:
ADU 1: Britain should build more nuclear power plants
ADU 2: Nuclear power is good for the environment

ADUs predicted by LLM:
ADU 1: Nuclear power plants should be built in Britain
ADU 2: Nuclear power is expensive
ADU 3: Nuclear power is good for the environment

To illustrate this alignment problem, consider the simple example in Example 3.2, where the gold-standard graph contains only two ADUs. An LLM model has predicted its own set of three text spans, and we must construct a one-to-one mapping between predicted and gold ADUs such that the most similar pairs of ADUs are aligned.

This alignment problem has been addressed in different ways in the literature. For example, Eger et al. (2017) rely on a simple word-overlap metric to match predicted and gold ADUs. In Chapter 7, Section 7.2.3, we discuss this issue in more detail, as in that setting the preprocessing and alignment procedure must be handled explicitly.

In this chapter, we are concerned with formally defining evaluation metrics over argumentation graphs, rather than with the problem of ADU span matching itself. Accordingly, we assume that a labelling function has been defined and applied to the predicted graph, yielding labels for predicted ADUs that can be compared against gold labels. Concretely, we assume that all ADUs in the set of gold ADUs, $ADUs_{\text{true}}$, have unique labels assigned by a gold labelling function label_T. Predicted ADUs in $ADUs_{\text{pred}}$ are then assigned labels via a matching function $\text{match}(p, t)$, which aligns a predicted ADU p with a gold ADU t. The predicted labelling function label_P is defined as follows:

$$\text{label}_P(p) = \begin{cases} \text{label}_T(t) & \text{if match}(p,t) \text{ for some } t \in ADUs_{\text{true}} \\ \text{new\_label}(p) & \text{otherwise} \end{cases}$$

To reiterate, the nature of the matching function differs across alternative approaches to the problem. Below, we move on to defining the three individual metrics in turn.

3.2.3 Definitions of the Metrics

Component-F1 (C-F1) calculates the correctness of the classification of different text spans into argumentative components (ACs or ADUs). Every span from the original text can be placed into one of four categories: MajorClaim, Claim, Premise or O (Outside the ADU/ non-argumentative). Following S&G, we calculate C-F1 as the Macro-average of the F1-Scores for each of these three categories {MC, C, P} (note that O is excluded from the evaluation).

Recall that F1-Score can be calculated using counts of true positives (TP), false positives (FP) and false negatives (FN) as follows:

$$F1 = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{3.1}$$

Let $G = (V, E)$ be a graph where V is the set of nodes and E is the set of edges. We first define two functions: $Label(G, u)$ which returns the label of a node u in a graph G, and $Type(G, u)$, which returns the ADU type of node u in graph G. We can then define a new function that returns the set of all labels for nodes of a given type (such as Claim) in a graph, as follows:

$$\text{ClaimNodes}(G) = \{\text{Label}(G, u) \mid u \in V, \text{Type}(G, u) = \text{“Claim”}\} \tag{3.2}$$

We can then calculate the number of True Positives, False Positives, and False Negatives as follows:

$$TP = |ClaimNodes(G_{\text{true}}) \cap ClaimNodes(G_{\text{pred}})| \tag{3.3}$$

$$FP = |ClaimNodes(G_{\text{pred}}) - ClaimNodes(G_{\text{true}})| \tag{3.4}$$

$$FN = |ClaimNodes(G_{\text{true}}) - ClaimNodes(G_{\text{pred}})| \tag{3.5}$$

We substitute the values from Eqs. 3.3 – 3.5 into Equation 3.1 to calculate the F1 score for Claim nodes, then use the same technique to calculate the F1 for MajorClaim nodes and Premise nodes. C-F1 is then calculated as the macro-average of these three scores.

Relation-F1 (R-F1) measures the correctness of the structure of the constructed argument graph. R-F1 is calculated as the macro-average between two F1 scores: F1-Link and F1-NoLink. To calculate F1-Link, we first define a function Links(G) that returns a set of pairs of labels, representing the edges which appear in a graph:

$$\text{Links}(G) = \{\, (\text{label}(G,u), \text{label}(G,v)) \mid (u,v) \in E \,\} \tag{3.6}$$

We can then calculate the number of TPs, FPs and FNs by substituting the Links function for the ClaimNodes function in Eqs. 3.3 – 3.5. For F1-NoLink, we define a function NoLinks, which returns all pairs of labels for nodes in the graph between which there is *no* link:

$$\text{NoLinks}(G) = \{\, (\text{label}(G,u), \text{label}(G,v)) \mid (u,v) \in V \times V,\ u \neq v,\ (u,v) \notin E \,\} \tag{3.7}$$

Analogously to our calculation of F1-Link, we substitute the NoLinks function for the ClaimNodes function in Eqs. 3.3 – 3.5. Relation-F1 is then calculated as the macro-average of F1-Link and F1-NoLink.

Relation Type-F1 (T-F1) evaluates the correctness of the edge labels. In the AAEC corpus, these can be in the set { *For*, *Against*, *Support*, *Attack*}, but following other authors (Eger et al. 2017; Kawarada et al. 2024, etc.) we reduce the set to {*Support*, *Attack*}, mapping *For* to *Support* and *Against* to *Attack*. T-F1 is therefore calculated as the macro-average of F1 scores for the two classes *Support* and *Attack*.

We again calculate F1 score by using Eqs. 3.3 – 3.5 to obtain the values to plug into Equation 3.1, except we substitute “Link” for the following functions for the “support” and “attack” classes respectively:

$$\text{SupportLinks}(G) = \{ (\text{label}(G,u), \text{label}(G,v)) \mid (u,v) \in E_{support} \} \tag{3.8}$$

$$\text{AttackLinks}(G) = \{ (\text{label}(G,u), \text{label}(G,v)) \mid (u,v) \in E_{attack} \} \tag{3.9}$$

where $E_{support}$ and E_{attack} are the sets of edges in G corresponding to *Support* and *Attack* relations, respectively.

3.2.4 Criticisms of ASP Metrics

One major aspect of the standard approach to ASP presented above can be criticised, namely, the proposal of ACTC as a separate subtask in its own right. All the main approaches to Argument Mapping (described in Section 2.3) present Argument Component types as something relational, i.e. the proposition contained in an AC is only a “premise" in relation to a conclusion that it supports, and indeed the same proposition could function both as a premise and as a conclusion in relation to different components. Therefore, this subtask, and its associated metric, are somewhat redundant.

It would therefore suffice to ignore C-F1 and use only the metrics measuring link identification and link type classification to evaluate ASP model performance. Nevertheless, when we address this task in Chapter 7, we will use all three standard metrics, in order to compare the performance of our techniques to previous work.

There is also another, more minor issue with how R-F1 is commonly calculated for the AAEC. Since this task is a binary classification task, (link vs. no

link) the use of two categories in the evaluation metric is redundant. Furthermore, since this is a highly unbalanced classification task, with the majority of node-pairs belonging to the node-link task, this will result in an inflated macro-F1 score, since the F1-score for the no-link category will almost always be very high. This can lead to an overly optimistic impression of performance for an uninformed reader; an approach which would allow a more realistic appraisal of performance would be to report the F1 score of the minority class only, i.e. F1-link.

3.3 ASP Corpora

In this section, we focus on corpora that other researchers have used for Argument Structure Parsing. All the corpora described below are in English unless otherwise specified.

The AAEC or Argument Annotated Essays corpus (Stab and Gurevych, 2014a) (also sometimes called PEC or Persuasive Essays Corpus) is among the earliest and most-widely used corpora for this task. The AAEC contains a number of essays from high school learners of English. It is notable for the regular structure of the annotations in the corpus. All essays are tree structured, with a single main claim appearing at the root of the tree (a MajorClaim). Each paragraph then consists of a subtree whose root is a claim that attacks or supports the MajorClaim.

The Microtext Corpus (MTC) (Stede et al., 2016) is a small corpus of 112 argumentative texts in English and German, deliberately composed by the authors to showcase argument structures. While small, it makes extensive use of features not shown in most other ASP corpora, including undercutting attacks.

Consumer Debt Collection Practices (CDCP) (Park and Cardie, 2018) is a dialogical corpus in the domain of eRulemaking, i.e. discussion of proposed laws which takes place on the internet, with the aim of collecting feedback for lawmakers. They use a domain specific annotation scheme which has four categories of EDU types relevant to the field of eRulemaking: Fact, Value (judgement), Policy, and Testimony. All links available are "support" rather than "attack", and there are two separate types of support relation annotated: "reason" and "evidence". This corpus is unusual among ASP corpora because in its annotation scheme, arguments do not necessarily form trees, but rather directed graphs. Non-tree graphs appear in their corpus when, for example, a single Fact is used as evidence for two different Policies.

The ECHR (Poudyal et al., 2020) is a corpus containing judgements from the European Court of Human Rights. It is annotated according to a simple scheme: the only two ADU types are "Premise" and "Conclusion", and the only available relation is "support".

Argument Annotated SciDTB (Accuosto and Saggion, 2019) contains annotations of scientific abstracts, annotated with a range of ADU types: proposal, assertion, result, observation, means, description, which can be linked together with "support" or "attack" edges.

ICNALE-AS (Putra et al., 2022) is a dataset of 434 English-as-a-foreign-language learner essays, which contain a somewhat similar tree structure to the AAEC, but additionally contain annotations of "restatement" and "detail" relations as well as "support" and "attack".

In addition, datasets annotated using Inference Anchoring Theory (IAT), which we have described in Section 2.3, can also be considered ASP datasets, since they contain an argument structure layer in addition to a rhetorical layer. Two of the most important of these are QT30 (Hautli-Janisz et al., 2022) and

US2016 (Visser et al., 2019), which are both large-scale (̃280,000 words for QT30, and 17,190 for US2016) and contain rich annotation schemes for both argumentative and rhetorical relations. Due to this added complexity, however, they are not normally tackled with the same techniques as are used for the other corpora that have been described in this subsection. The following subsection (Section 3.4, **Approaches to Argument Structure Parsing**) will focus on mainly on techniques that have been used on more typical ASP corpora, of which the paradigmatic example, and the one on which the greatest volume of experimental work has been carried out, is the AAEC. However, we will return to addressing work on IAT corpora in more detail in Section 2.5 on Dialogic Argument Mining.

3.4 Approaches to Argument Structure Parsing

In this section, we will discuss models that have been developed for ASP. The majority of ASP methods, since the earliest work have been ADU-level models (also sometimes called Argument Component (AC) level or proposition-level in the literature). These are models which assume ADU segmentation as a prior step, and usually use oracle segmentations in their experiments. However, there has been another strand of work in the literature since Eger et al. (2017) which is normally referred to as end-to-end. These are models which do not rely on prior segmentation into ADUs, such as token-level tagging models. We will first describe the ADU level models and then end-to-end models.

3.4.1 ADU-Level Models

Earlier methods for carrying out Argument Structure Parsing used classifiers that relied on hand-crafted feature sets. For example, the models used in Stab and Gurevych (2017) or Afantenos et al. (2018) rely on a large number of features for ACTC, including the length of ADUs, the presence of certain words or phrases that might indicate the ADU type (e.g. "in conclusion"), as well as word embedding features generated using the Word2Vec model (Mikolov et al., 2013).

Potash et al. (2016) was an early paper applying neural models to the ASP task. A Pointer network is used (Vinyals et al., 2015) a type of sequence-to-sequence model with attention, which can be used to decode sequences over the encoding inputs. In this case, the input to the model is a sequence of ADUs, and we want to decode the relations between them. This model used less feature engineering than the previous models, using pooled GloVe representations (Pennington et al., 2014), as well as a bag-of-words representation and a positional encoding of the ADU. Kuribayashi et al. (2019) take a similar approach, encoding each ADU using various different types of span representation based on both GloVe and Word2Vec. Instead of using a pointer network for the Link Identification portion of the task, the authors took the logits returned by a binary link/no-link classification head as graph edge weights, and used a maximum spanning tree algorithm (Tarjan, 1977) to decode the most likely output tree.

Niculae et al. (2017) introduce an alternative factor-graph based model, and they experiment with using both RNNs and SVMs to learn the parameters of the model. The factor graph models allow the explicit encoding of latent structural relationships, such as that between a parent node and its grandparent, and also allow the enforcement of arbitrary constraints on the resulting output

graphs such as transitivity.

Morio et al. (2020) developed a model that applied biaffine attention (Dozat, 2016), a special attention mechanism developed for dependency parsing, to the task of ASP. In contrast to previous neural models, this model is capable of decoding non-tree argument graphs, such as those in the CDCP corpus.

Bao et al. (2021) is, to our knowledge, the first study to apply a transformer-based pretrained Language model, BERT (Devlin et al., 2018) to the task of ASP. This paper is also notable for taking a fundamentally different approach to the ASP task: a neural transition-based parser is used (Chen and Manning, 2014). The important distinction is that instead of directly predicting relations between pairs of ADUs as in previous models, the model must incrementally construct a graph by predicting an *action* at each step, e.g. the LEFT-ARC action, which assigns a left-pointing arc in the argument structure graph. This technique is advantageous due to having a time complexity (with respect to the number of ADUs in the input) of $O(n)$ at inference time rather than $O(n^2)$, as is the case with other models which compare ADUs pairwise.

Gemechu and Reed (2025) make several innovations in their work, the most important of which is likely the use of Graph Neural Networks (GNNs) within a neural architecture, to represent argument structure. Their model uses LLM layers as inputs to a hybrid structure which stacks a GNN and multiple attention layers. As well as this novel neural network structure, another contribution of this work is adding auxiliary objectives to the model training. More specifically, in addition to commonly-used classification heads for the ACTC and ARTC tasks, the model adds two heads for classifying "local" and "global" argument structures according to a fixed set of categories. These tasks are not used in parsing the output, but the authors find that they improve model performance, and are therefore likely useful for inducing the model to learn to represent graph structures.

3.4.2 End-to-End Models

All the models in the section above relied on a prior step of segmenting the input into ADUs before carrying out ACTC, ARI and ARTC subtasks. In contrast, the methods described in this section are all "end-to-end", that is, they transform the text input into an argument graph without this prior segmentation step.

Eger et al. (2017) were the first to apply neural end-to-end models to ASP, and they tested five alternative models for the task, with two main model types/ frameworks. These are (a) token-level tagging, and (b) token-level dependency parsing. They find that their token-level tagging model, based on BiLSTM-CRF-CNNs, performed best for relation detection, while their dependency parsing model, based on LSTMs, performed best for classifying components.

Ye and Teufel (2021) improve on Eger et al.'s (2017) parsing model by using a neural model with a BERT encoder and biaffine attention instead of an LSTM, as well as altering the labelling of relations in the dependency parsing output.

Kawarada et al. (2024) were the first to attempt the ASP task as "text-to-text translation". In this framework, the entire task is carried out end-to-end as a "translation" between a raw input text (for example, an argumentative essay) and an output text which has been annotated using a markup language representing the argument structure. Using the Flan-T5-XXL model, they find that this technique outperforms older models in the end-to-end setting.

3.5 Argument Summary Parsing in Dialogue

ASP in dialogue adds an additional layer of complexity due to the fact that one must choose how to encode information about speakers and their stances in either the input or output representations. This choice depends to a degree on the aims of the task; if the goal is simply to create a speaker-neutral representation of all arguments in a debate (for example, to evaluate the arguments for and against a specific point without any reference to who said them), then one could treat the task more or less identically to monological ASP; this is the approach taken in the CDCP dataset (described in Section 3.3).

However, in other work, annotation schemes have been chosen are chosen which do require models to pay attention to who said what. The most prominent of these annotation schemes is Inference Anchoring Theory, and we will now move on to describe ASP work that uses this formalism.

3.5.1 ASP using Inference Anchoring Theory

One important strand of research on ASP in dialogue is that involving IAT (Inference Anchoring Theory), an approach to jointly mapping/ annotating argument and dialogue structures which we have described in Section 2.3.2.

Due to the complexity of the task of producing IAT graphs from text, most work have concentrated on only part of the task. Below, we will first look at research which has looked on individual subtasks (i.e. ARI or ARTC) and then at works which have attempted to build complete ASP systems.

Work on individual subtasks A number of works on IAT corpora have tried to solve one or more of the individual subtasks in isolation. Gemechu and Reed (2019) are the first to address two different subtasks — ARI and ARTC — on the US2016 dataset (Visser et al., 2019). To detect relations, a pipeline of classifiers is used first to identify different elements of propositions within the two ADUs, and then to detect whether the two contain a contradiction. Various classifier types are tested, including naive bayes, SVMs and CNNs, using continuous bag-of-words features.

Ruiz-Dolz et al. (2021a) tackle the ARTC task, also using the US2016 corpus, and show that improvements in performance are possible by using LLM-based classifiers. Kikteva et al. (2023) took a similar approach to the ARTC task, using the QT30 corpus instead, and found that performance could be improved somewhat by adding the "context" of the propositions (namely, their corresponding locutions) as input to an LLM.

Gemechu and Reed (2024) further improve ARTC performance with a more complex approach to ARTC, which incorporates domain knowledge from Wikipedia into this task. A pipeline approach is taken, in which firstly key words about the topic of the argument are identified in both ADUs; next, a graph-based Wikipedia scraping technique is used to find sentences discussing these keywords; and finally these sentences are used to generate an embedding suitable as an input to a neural network.

Work on the full ASP task Gemechu and Reed (2025) appears to be the first work showing a model which is applied to the full task of ASP on an IAT dataset, albeit only on the argumentative layer. We have covered their model in Section 3.4.1; it achieves state-of-the-art results on both QT30 and US2016, as well as several monological ASP corpora.

A number of models have also been developed in response to the DialAM-

2024 shared task (Ruiz-Dolz et al., 2024). In this task, the input assumes that just two parts of the pipeline have already been completed successfully; (1) the segmentation of the input dialogue into ADUs and (2) the extraction of propositions from the ADUs. Therefore, the input to the DialAM task consists of two sets (See Chapter 2, Section 2.3.2 for an explanation of these aspects of the IAT formalism):

1. A set of locutions, linked together in order by "default transition" nodes.
2. A set of argumentative proposition nodes, lacking any links between them

The models must also predict two different types of outputs: (1) the relation nodes between the argumentative propositions, and (2) the illocutionary relation nodes. These two tasks are not independent, because as Figure 2.3 shows, some of the illocutionary relations link to argumentative proposition link nodes.

All of the globally best-performing systems at this task (Binder et al., 2024; Chaixanien et al., 2024; Wu et al., 2024) used fine-tuned LLM classifiers; the best performing model, Binder et al. (2024), used normalisation techniques to allow a single model to be trained to classify multiple different link types.

3.6 Conclusion

In conclusion, research into ASP has proceeded along two main strands; monological ASP, in which the output tends to be formalised either as tree structures or Freeman-style argument graphs, and dialogical ASP, in which the most common formalism is Inference Anchoring Theory. A large amount of research has been carried out into algorithms for monological ASP, including models

that operating either fully end-to-end or using pre-segmented ADUs as the input. Research on the ASP using IAT has to date mostly concentrated on small subtasks; however research into tackling the entire task end-to-end has begun to appear with in such work as Gemechu and Reed (2025) and Ruiz-Dolz et al. (2024).

Chapter 4

Reasoning Marker Prediction

4.1 Introduction

In this chapter, we introduce a novel Argument Mining task which we refer to as Reasoning Marker Prediction (RMPred). Reasoning Markers (RMs) are a subcategory of Discourse Markers which have a specific function within argumentative text. The RM category includes words and phrases such as "because", "therefore" or "in conclusion" which can be used to structure an argumentative piece of text, acting as the "glue" to hold a text together and make it more intelligible. In our proposed RMPred task, a system takes as input a textual context (consisting of text preceding and/or following the location that the RM may appear) and must predict whether an RM appears in this context or not.

We believe that this task is important as it can form part of a pipeline approach for generating a textual summary from an argumentative text. The proposed pipeline would involve first generating an argument graph, then selecting relevant Argument Components (ACs) from this graph, which could finally be transformed into a fluent text by adding Reasoning Markers.

We release a dataset for for training and evaluating systems designed to carry out the RMPred task, which we produce using the Argument Anno-

tated Essays Corpus (AAEC) (Stab and Gurevych, 2014a) as a data source. We additionally fine-tune multiple Large Language Models (LLMs) for the task of RMPred. These models outperform a random baseline, with the best performing model being an off-the-shelf BERT model, which achieved an F1-score of 0.69.

4.2 Motivations

Our motivation for investigating the RMPred task is that it may have utility within a pipeline approach for carrying out summarisation.

As stated in the introductory chapter (Chapter 1, Section 1.2), our overall goal in this thesis is the investigation of two different types of summaries of arguments: graphical summaries and textual summaries. One way in which a textual summary could plausibly be generated is through a pipeline approach, *via* first using Argument Mining to generate a graphical summary, and then using this graph to produce a textual summary. This type of approach was proposed in Barker and Gaizauskas (2016) for the summarisation of dialogues (we summarise their proposed algorithm in more detail in Chapter 5, Section 5.5).

Elaborating on this idea, our conception of an algorithm for generating a textual summary of an argument would be a pipeline with three steps. Starting with a single argumentative essay as an input, it would do the following:

1. Use Argument Structure Parsing (ASP) to generate an Argument Graph representing all Argumentative Discourse Units (ADUs), and the relations between them (e.g. "attack", "support") which are present in the essay.

2. Extract the n most important Argument Components from the Argument

Graph (Where n is a threshold controlling the length of the resulting summary[8]). As part of the same step, order the components based on the relations extracted in the previous step (for example, premises should be placed adjacent to the conclusions that they support).

3. Combine the Argument Components extracted in the previous step into a readable summary, adding "shell language" including Reasoning Markers.

In this chapter, we focus entirely on Step 3. This final step is crucial for producing a text that is coherent and easy to read; discourse markers in text have been found to increase reading comprehension scores experimentally (Degand and Sanders, 2002).

Note that Step 1 of this proposed pipeline, Argument Structure Parsing, has been extensively investigated in Argument Mining literature (see our review of ASP in Chapter 3). Step 2, that of designing a function to select the most relevant components from an argument graph to produce a textual summary, has not to our knowledge been addressed by previous research; however, it is beyond the scope of this chapter.

4.3 Background

In this section, we first give the definition of the term "reasoning marker" as used in this chapter, and then go on to review work on tasks related to Reasoning Marker Prediction.

[8]Choosing the value of n might not be entirely straightforward; for example, we might want the length of the summary to depend in part on the length of the text to be summarised. However, we would expect the choice of this value to vary between applications, and we will not address this issue in any more detail here.

4.3.1 Defining Reasoning Markers

Reasoning markers (RMs) are a proper subset of discourse markers (DMs), i.e. those words or phrases used in the organisation of a spoken or written text — in the case of RMs, argumentative text specifically. The notion of DMs, and the utility of DMs within argumentative text, were both first introduced in Schiffrin (1987). An extensive taxonomy of DMs was produced by Knott (1996), and further developed by (Oates, 2000). However, these taxonomies lack a specific “Reasoning Marker" category. Williams (2018) seems to be the first to have used the phrase “reasoning marker". This study found that the presence of RMs is argued to be positively correlated with the academic trustworthiness of a text.

Sets of discourse markers that we would define as RMs have been used previously in a range of Argument Mining tasks, including as a feature for the identification of claims and premises, and the relations between them (Stab and Gurevych, 2014a; Eckle-Kohler et al., 2015; Lawrence and Reed, 2015). All of these AM tasks have used a pre-defined closed list of RMs rather than relying on a prior reasoning marker detection step.

Some uncontroversial examples of RMs are “because” and “therefore”, which are used to indicate a logical connection between a conclusion and one or more premises. The term Reasoning Marker excludes, however, those Discourse Markers which would typically be found in narrative text, such as “once upon a time”, “eventually”, or “suddenly”.

Beyond this specification that RMs are Discourse Markers used in argumentation, we will not attempt provide a rigorous definition of the notion of “reasoning marker” since we believe this is a complex linguistic problem beyond the scope of this work. Categories of discourse marker are notoriously difficult to define, and may be best conceptualized as “family resemblance”

categories rather than categories definable by a list of formal features (Bordería, 2006). Instead we sidestep the issue by assuming that whatever linguistic material can be used to connect together Argument Components counts as a Reasoning Marker. For our purposes we consider this definition satisfactory, since we are not aiming at formal linguistic correctness but rather at generating a coherent and readable text.

4.3.2 Related Work

Several authors have addressed tasks related to the one proposed in this chapter; however, due to their differing goals, none of the datasets that they produce are directly applicable to this task. Rocha et al. (2023) look at the prediction of Discourse Markers within Argument Mining, but with a different focus to ours. Their goal is to "augment" texts by adding discourse markers to them as a first stage in an Argument Mining pipeline. Their intuition is that by making implicit discourse markers explicit, they can enhance the clarity of the data fed into a "downstream" AM system, and thereby improve the downstream system" performance. The dataset that they produce is not quite suitable for our task, since we are not interested in making *every* discourse relation in the text explicit, but rather in inserting RMs in a way that will produce a fluent and readable textual summary.

A similar task to this one was proposed by Patterson and Kehler (2013), who used data from the Penn Discourse Treebank (Pennington et al., 2014) to construct a dataset of clause pairs with either an "explicit" or "implicit" discourse marker. For instance, they give the example of "Max will visit Australia this summer because his father is turning 65." (two clauses connected by the *explicit* DM, because) and "Max will visit Australia this summer. His father is turning 65." (the same clauses connected by an *implicit* DM). The task is then, given a pair of clauses, to predict whether the DM will be realised explicitly or

implicitly. This differs from our task, in that we look at ADU boundaries rather than clause boundaries, and datapoints which are in the "no RM" category need not necessarily contain an implict RM (in some cases there could simply be no discourse relation, implicit or explicit, between adjacent ADUs).

Malmi et al. (2017) propose yet another related task, which they refer to as "discourse marker prediction", but we will refer to here as "discourse marker *generation*" to avoid ambiguity. In this task, instead of simply predicting whether or not a discourse marker occurs in a given context, a model must generate the actual discourse marker which should appear in a given position. Using a list of 19 common discourse connectives plus a "no connective" category, they treat DM generation as a 20-class classification problem, for which they train a neural network model. Unlike our work in this chapter, they focus on predicting only connectives in the sentence-initial position.

4.4 Task Definition

In this section, we define the Reasoning Marker Prediction (RMPred) task, which involves predicting whether or not a reasoning marker occurs in a text given the surrounding context. As an input, we use a full essay represented as a string, with a single `[RM]` special token indicating a location where a reasoning marker may or may not appear. This is a binary classification task, and therefore has two possible output labels, "True" indicating that an RM is predicted to be present in the location of the `[RM]` special token, and "False", indicating that no RM is predicted to occur in that position.

Example 4.1 illustrates a single datapoint from the corpus, including the input and output. Note that we have removed all uppercase letters and punctuation in the original text, as this could provide a language model with spurious

clues about where an RM could be present - for example, a model could learn to never predict an RM before a capital letter indicating the start of a sentence. These features would not be present in the real use case that we envision for this task, i.e. the bottom-up construction of a summary using ADUs.

Our task requires the model to predict the presence/ absence of only one reasoning marker at a time. Therefore, our dataset contains multiple duplicate copies of the same essays (one for every "potential RM" location in each given essay). We ensure that these duplicate essays do not appear across train/ test/ validation splits, to avoid evaluating on data seen during training.

Example 4.1: Input/output schema for RM prediction task.

Input: "furthermore investing in art could bring employment opportunities and could end in return of capital occasionally [RM] the investment could be paid..."
Output: "False"

4.5 Research Questions

In this chapter, we will investigate two research questions related to the RMPred task defined in the previous section.

RQ 4.1 — How well do LLMs perform on the RMPred Task? The first research question is to test how well Large Language Models can work for the RMPred task that we propose in this chapter. We investigate two LLMs which were state-of-the-art at the time that the experiments were carried out: BERT (Devlin et al., 2018) and T5 (Raffel et al., 2020).

RQ 4.2 — How does augmenting the data with AC types affect LLM performance? The second RQ asks whether or not an LLM applied to this task for this task can benefit from added information about the Argument Component (AC) types being available during fine-tuning and testing. Specifically, we look at augmenting the text itself with "special tokens" that are prepended to AC text spans and indicate the AC category that each span belongs to.

Our intuition for doing so is that the AC type may be correlated with the discourse indicator chosen in some cases (see Lawrence and Reed 2015); one straightforward example of a pattern we may expect to occur commonly is "[claim] because [premise]". If adding AC information were to provide a benefit, that would suggest that a pipeline approach may be beneficial for this task, wherein the type of AC spans is first automatically classified prior to doing RM prediction.

4.6 Dataset Generation

In this section, we describe the process by which we generated the dataset used in this chapter. We use a simple heuristic method to identify Reasoning Markers in an already existing corpus, taking advantage of existing annotations. In Section 4.6.1, we describe briefly the existing annotation scheme of this corpus, and in Section 4.6.2, we describe how we produce the new corpus based on these annotations.

4.6.1 Argument Annotated Essays Corpus - Existing Annotation Scheme

The corpus which we choose to use for the extraction of Reasoning Markers is the Argument Annotated Essays Corpus (AAEC) (Stab and Gurevych, 2017). The AAEC (which we describe in more detail in Chapter 3, Section 3.1) is a corpus annotated for the task of Argument Structure Parsing, consisting of 402 persuasive essays on a variety of controversial topics.

Example 4.2: An illustration of the BIO-tagging scheme used in the AAEC.

Furthermore	,	investing	in	art	could	bring	employment		
O	O	B-Claim	I-Claim	I-Claim	I-Claim	I-Claim	I-Claim		
opportunities	and	could	end	in	return	of	capital	occasionally	
I-Claim	I-Claim	I-Claim	I-Claim	I-Claim	I-Claim	I-Claim	I-Claim	I-Claim	
.	The	investment	could	be	paid	back	through	the	
O	B-Premise	I-Premise	I-Premise	I-Premise	I-Premise	I-Premise	I-Premise	I-Premise	
values	of	the	created	works	of	art	which	as	
I-Premise	I-Premise	I-Premise	I-Premise	I-Premise	I-Premise	I-Premise	I-Premise	I-Premise	
a	matter	of	fact	should	be	considered	as	national	
I-Premise	I-Premise	I-Premise	I-Premise	I-Premise	I-Premise	I-Premise	I-Premise	I-Premise	
possessions	.	To	sum	up	,	not	only	could	investing	in
I-Premise	O	O	O	O	O	B-Premise	I-Premise	I-Premise	I-Premise	I-Premise
art	be	considered	as	wasting	money	at	any	kind		
I-Premise	I-Premise	I-Premise	I-Premise	I-Premise	I-Premise	I-Premise	I-Premise	I-Premise		
,	but	also	it	would	enriches	the	culture	of	the	
O	O	O	B-Premise	I-Premise	I-Premise	I-Premise	I-Premise	I-Premise	I-Premise	
society	.									
I-Premise	O									

Example 4.2 illustrates the annotation scheme of the AAEC. The corpus is labelled with ADU (Argumentative Discourse Unit) spans, of which there are three types: MajorClaim, Claim and Premise (this is explained in more detail in Chapter 3, Section 3.1). The text is divided into spans using a BIO-tagging scheme , in which each token is labelled as being at the Beginning (B), Inside (I), or Outside (O) of an ADU span. The tag for each token forming part of a span is a combination of the B/ I label and the ADU type label, hence the

full set of possible tags is {B-MajorClaim, I-MajorClaim, B-Claim, I-Claim, B-Premise, I-Premise, O }.

Example 4.3: The essay fragment from Example 4.2

[Furthermore , RM] [investing in art could bring employment opportunities and could end in return of capital occasionally CLAIM] [. NO RM] [The investment could be paid back through the values of the created works of art which as a matter of fact should be considered as national possessions PREMISE] [. To sum up , RM] [not only could investing in art be considered as wasting money at any kind PREMISE] [, but also RM] [it would enriches the culture of the society PREMISE] [NO RM]

4.6.2 Inferring Reasoning Markers

To generate our dataset, we must both extract RMs, and identify locations where RMs could plausibly be inserted but do not appear. The latter category of locations are those marked [NO RM] in Figure 4.3).

We observe that the vast majority of these O-labelled sentence fragments either contain just a punctuation mark or can be considered as containing an RM. This should not be surprising for two reasons: (1) as just outlined, all of these sentence fragments come attached to ACs; (2) the essays originate from `essayforum.com`, a website consisting mostly of essays composed by high-school students or learners of English as a second language – educational contexts where students are rewarded for including RMs within texts.

Making use of this observation, we use a three stage pipeline to identify RMs and [NO RM] locations:

1. Split the text into sentences, i.e. perform sentence tokenisation.

2. Select sentences containing at least one Argument Component (AC), as

well as one or more O tags.

3. Within each selected sentence, iterate over all adjacent groups of O tagged-tokens within each sentence. If the group consists of just a punctuation mark, it will be labelled as a [NO RM] location. Otherwise, if it contains word tokens, it will be classified as an [RM].

4.6.3 Corpus Contents

	RM	**No RM**	**Total**
Train	2726	2550	5276
Validation	346	287	633
Test	802	715	1517
Total	3874	3552	7426

Table 4.1: Numbers of samples found in train, validation and test sets.

Reasoning Marker	**Frequency in Corpus**
“because”	195
“for example”	178
“therefore”	137
“however”	110
“moreover”	104

Table 4.2: The five most common reasoning markers appearing in AAEC

Our processed version of AAEC contains a total of 7426 “potential RM” datapoints. The data is evenly balanced between the classes RM/No RM, as can be seen in Table 4.1.

The corpus contains a total of 1264 reasoning marker types. While this number seems large, it is somewhat artificially inflated by a number of minor

variations on what are semantically very similar RM phrases, such as "To conclude, I definitely feel that", "To conclude, I strongly believe that", "To conclude, I want to say that", and a number of similar examples. Shorter RMs are also much more common than longer RMs, as shown in Table 4.2 and Figure 4.1. 39.8% of all RMs are only a single token long. This concurs with Zipf's brevity law [9] , which states that more frequently occuring lexical items tend to be shorter (Zipf, 1949, pp. 64–66).

The classification of some of the longer segments as RMs is somewhat dubious. For example, the following 25-token phrase would not be typically classified linguistically as a RM, but it seems to fulfil a similar function in context:

> "In conclusion, after analyzing the pros and cons of advertising, both of the views have strong support, but it is felt that... << *conclusion* >>"

However, we refrain from filtering out discourse markers using linguistic criteria, since we treat this as an engineering task and mainly aim to add in appropriate connective material between ACs, whether or not they count as RMs in any defensible linguistic sense.

4.6.4 The Corpus Processing Script

We release a script via our repository (`github.com/acidrobin/reasoning_marker_prediction`) which takes in the data provided in the AAEC repository (`github.com/UKPLab/acl2017-neural_end2end_am`) and converts it into valid input for a language model. We describe the format of this input in Section 4.4.

[9] Not to be confused with Zipf's better-known eponymous law which relates a word's frequency to its rank (Zipf, 1949, pp. 25–30)

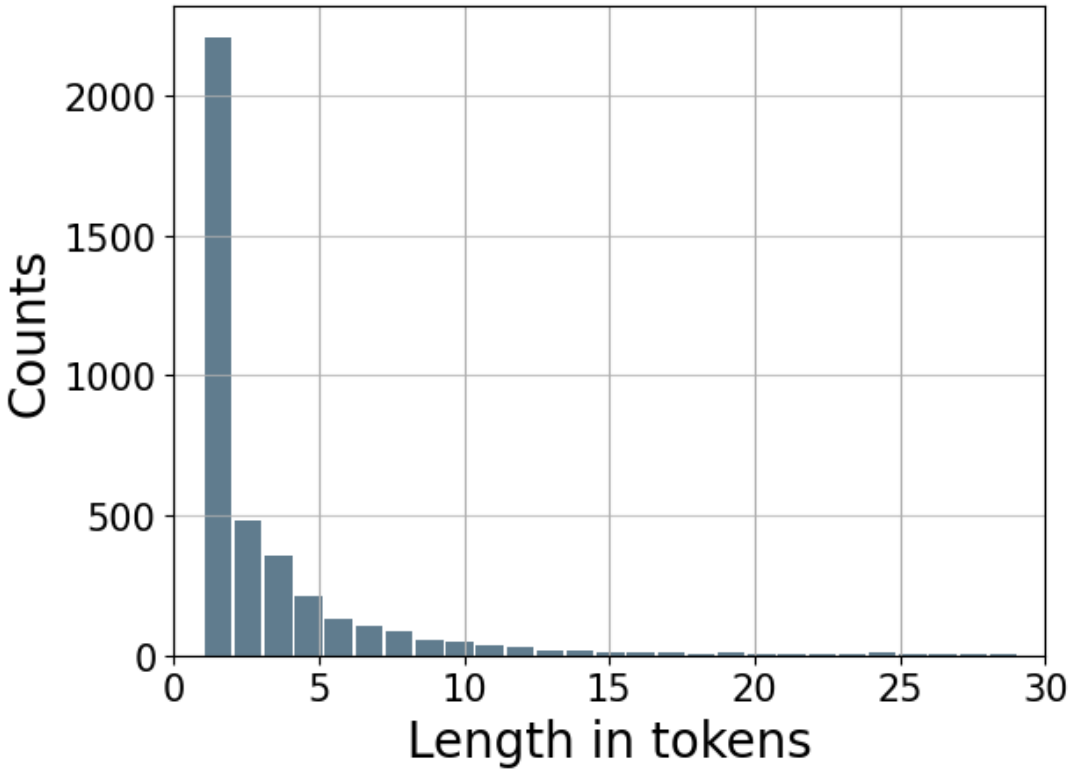


Figure 4.1: Counts of RM tokens in AAEC by length in whole-word tokens

4.7 Experiments

In order to set a benchmark for this task, we carried out experiments by fine-tuning two different pretrained LLMs: BERT (Devlin et al., 2018) and T5 (Raffel et al., 2020). The models are compared to a random baseline, i.e. a model which randomly chooses either "RM" or "No RM" with a uniform probability of 0.5 for each. This reflects the near-even distribution of the two categories in the dataset.

Below, we will firstly describe the models used and their fine-tuning process, followed by a description of how we augmented the data with Argument Components to address RQ 4.2.

4.7.1 Models and Fine-Tuning

Hyperparameter	BERT	T5
Model Name on huggingface.co	`bert-base-uncased`	`t5-base`
Learning Rate	$1e^{-5}$	$5e^{-6}$
Epochs (Early Stop)	5 (out of 8)	5 (out of 8)
Optimiser	Adam (Kingma and Ba, 2014)	
LR Grid Search	$\{1e^{-4}, 5e^{-5}, 1e^{-5}, 5e^{-6}, 1e^{-6}\}$	

Table 4.3: Fine-tuning hyperparameters for BERT and T5 models

The BERT and T5 models share fundamental similarities in that they are transformer models (Vaswani et al., 2017), however they differ in both the overall architecture and the specific pre-training regime that they follow (see Raffel et al. (2020) for details).

Implementations of the two models are taken from `huggingface.co` (Wolf et al., 2019). For both models, we used lower-cased text in the input, to prevent trivial classification using the case of the word following the potential RM location.

Due to its pretraining scheme, T5 benefits from a task-specific linguistic prompt being prepended to the input. We experimented with 3 options: no prompt, "True or False:", and "Is there a reasoning marker? True or False:". We found the prompt "True or False:" gave the best result.

For the tokenizers of both models, we add in a special [RM] token which indicates a potential reasoning marker position. For the +AC test condition, we add [claim], [majorclaim], and [premise] special tokens to the BERT tokenizer. Since the T5 model can generate free text, we constrained its output

at inference time, by restricting the model to only produce a single token, and further constraining that token to be from the set {“true”, “false”}, by setting the values of the logits corresponding to all other tokens to zero.

4.7.2 AC-Augmented Data

As described above in Section 4.5, one of our research questions is to investigate whether the performance of models can be improved by introducing information about Argument Component (AC) type to the training and test data.

Our procedure for augmenting the data was straightforward. Firstly, each AC type — Premise, Claim and MajorClaim — was encoded as a “special token”, i.e. an additional token outside of the vocabulary on which the LLMs have been pretrained, which is added to is to the tokenizer and assigned a new embedding vector that the model can potentially learn during fine-tuning. Each of these tokens was encoded as the name of the AC type in lower case within square brackets, i.e. [premise], [claim], [majorclaim].

We added these special tokens into the essay text by prepending them before each corresponding AC span in both training and test data. For example, a paragraph containing a Premise and Claim would have an input similar to “[premise] *premise text* [RM] [claim] *claim text*” – so that the [premise] and [claim] special tokens indicate the beginning of these components.

4.7.3 Details of Training Scheme

All of our BERT and T5 models were fine-tuned for 8 epochs, and early stopping was used, with the model stopping early if performance on the validation set failed to improve for 2 consecutive epochs. Due to the fact that all our data was lower-case, the uncased version of BERT was used, while the T5 model used was cased, since an uncased version has not been released. We used the uncased version of BERT base. We use the Adam optimizer (Kingma and Ba, 2014). The best learning rate is chosen by a grid search; for both models we explore the set $\{1e^{-4}, 5e^{-5}, 1e^{-5}, 5e^{-6}, 1e^{-6}\}$. For the BERT model, we found the optimal learning rate was $1e^{-5}$ and the model stopped after 5 epochs of fine-tuning. For T5, a learning rate of $5e^{-6}$ was optimal, and the model also stopped fine-tuning after 5 epochs.

4.8 Results

We evaluate our results using precision, recall and F1-score macro-averaged between the two classes. Predictably, the random baseline achieved a Macro-F1 score close to 0.5 (reflecting the even class balance in the test set).

Model Name	Precision	Recall	F1-Score
RandomBaseline	0.47	0.51	0.49
bert-base	**0.75**	**0.70**	**0.69**
bert-base+AC	0.73	0.66	0.64
t5-small	0.63	0.60	0.59
t5-small+AC	0.60	0.57	0.57

Table 4.4: Performance of the baseline and three models evaluated on the test set.

LLM Performance (RQ 4.1): Table 4.4 shows that the T5-small model underperformed compared to BERT. This result is somewhat surprising, due to the fact that T5-small is pretrained on vastly more data. However, it does have a lower number of parameters than BERT, and it has been shown that this model may struggle at fine-tuning on relatively lower-resourced tasks such as this one (Ulčar and Robnik-Šikonja, 2023).

Augmentation with AC Type (RQ 4.2): As Table 4.4 shows, the best-performing model was the "vanilla" bert-base-uncased. Contrary to what we had predicted, the +AC models, which were trained and tested on data which had special tokens added to indicate Argument Component types, underperformed compared to the model which lacked this additional data. This is unexpected given the known correlations which exist between AC types and reasoning markers. One possible explanation for this could be related to the pre-training of LLMs; LLMs are pretrained on enormous amounts of data which tend to be entirely in unstructured natural language format. Perhaps the model is unable to learn the meanings of the AC type special tokens using only the data in our relatively small fine-tuning set.

4.9 Conclusion

We have presented a new task, Reasoning Marker Prediction or RMPred, which involves predicting whether or not a Reasoning Marker will appear between two ADUs in an argumentative text. We believe this task can form part of a pipeline for generating argumentative text summaries. We have released a script that can be used to generate our derived corpus from AAEC, which supports this task. Additionally, we have shown it is possible to predict the presence or absence of an RM between two Argument Components at an above-

chance level. Our baseline scores show this is a challenging task, with much room for improvement.

4.10 Limitations and and Future Work

A natural extension of this task is not only to predict *that* an RM should occur but *what* the RM should be. In the future, we aim to work on using end-to-end models to generate an appropriate RM for a given context, instead of simply predicting whether or not an RM should appear.

Another aspect of this task which we have not explored is the sub-categorization of RMs. Multiple taxonomies of DMs have been developed that could be used for this task. See Knott's (1996) taxonomy, and the development in Oates (2000). However, it is likely that this would be a non-trivial task and require some expert labelling, due to the fact that there is not a one-to-one correspondence between DMs and their functions. A simple DM like "so" for example, has many different functions and can be used to provide justifications, for sequencing, or for expressing a purpose. Nonetheless, since, as noted above, there are many RMs in this dataset that are more-or-less interchangeable, it may be sufficient to predict the category that a potential RM should belong to rather than attempting to generate one directly.

Chapter 5

Argument Summary Graph Parsing

5.1 Introduction

In this chapter, we introduce the Argument Summary Graph Parsing (ASGP) task. This is our proposed task for generating a graphical summary from a dialogical text that contains argumentation, for example, an online news comment section. This chapter forms the background for the work that we go on to carry out on the topic of ASGP in Chapters 6, 8 and 9.

This chapter contains four content sections. First, in Section 5.2, **Background**, we provide background relative to the development of the ASGP task. Next, in Section 5.3, **The ASG Parsing Task**, we provide a full definition of the task. Next, in Section 5.4, **Evaluation**, we describe the metrics which we have proposed to evaluate ASGP, which we will use in subsequent chapters. Finally, in Section 5.5, **Applications of ASGs and Post-Processing**, we explain a few ways in which we might expect automatically-generated ASGs to be post-processed in practical applications.

5.2 Background

In this section, we first address our motivation for designing a new task (Section 5.2.1), next, how this task relates to the well-known task of Argument Structure Parsing which we reviewed in Chapter 3 (Section 5.2.2), and finally cover the corpora and data resources of relevance to this task (Section 5.2.3).

5.2.1 Motivation

Online comment threads have several attributes which make them less accessible to a casual reader who wants to quickly get an overview of the topic under discussion. We have discussed these factors at length in Chapter 1, Section 1.3. They include the fact that often they (a) are very long, (b) contain redundant comments, and (c) are not necessarily organised in a logical order.

We propose a logically organised, graphical representation for online comment threads. Specifically, we are interested in *argumentative* comment threads (that is, comment threads which contain a debate) and therefore propose a type of graph which can display the argumentative relations between comments (in our case "support" or "attack"). These structures are fundamentally the same as the argument maps found on `kialo.com` (Kialo, 2024), which themselves bear similarities to the argument mapping techniques developed in works such as Freeman (2011) and Budzynska et al. (2014b). Throughout the rest of this thesis, we will refer to these structures as Argument Summary Graphs or ASGs.

ASGs are designed to allow readers of the dialogue to more easily and quickly obtain a broad overview of the most important issues under discussion without the need to read a large number of online comments. This may be

useful for a general audience interested in knowing the opinions of the readership of a certain online forum or newspaper on a particular controversial topic. Additionally, this might benefit more specialised audiences who have an interest in the automatic analysis of public opinion such as journalists and social scientists. In order to have an intuition for how such graphs might be helpful, the reader can consult the manually-created interactive argument graphs on kialo.com (Kialo, 2024), which illustrate the type of interface which could be used to interact with an automatically-generated ASG.

5.2.2 Relation of Argument Summary Graph Parsing to Argument Structure Parsing

Argument Summary Graph Parsing is closely related to the established Argument Mining task of Argument Structure Parsing (ASP), which we have thoroughly reviewed in Chapter 3. This is the task of extracting all argumentative components from a text, and identifying the argumentative relations which obtain between them.

While in ASP, as in ASGP, we are interested in identifying argumentative links between different text spans, the spans in ASP tend to be at the level of an Argumentative Discourse Unit (see Section 2.3) instead of at the level of a whole comment, as we have proposed here. Additionally, in ASP, the output must contain labelled spans from the input, in contrast to our task, where the final tree consists of summaries of comments from the original text.

5.2.3 Existing Data Resources and Corpora

In this subsection, we will address the issue of available corpora and data that could be applied to ASGP. Firstly, we will look at already-existing corpora from related AM tasks and assess their utility for our purposes. Secondly, since we find that no existing corpora satisfy the requirements for training data for this task, we also briefly survey some online resources which could be used to generate ASGP corpora.

Annotated Corpora As we have discussed across Chapters 2 and 3, there are a number of corpora that have been produced for AM tasks similar to this one, but none, we argue, that address the specific niche of ASGP.

In Sections 2.4, 2.5, and 3.3, we have reviewed corpora designed to support Argument Summarisation, Dialogic AM, and Argument Structure Parsing, respectively. While there are already many corpora containing either annotated relations between comments/ utterances in a dialogue, or summaries of arguments, there are none that contain both (1) informative per-comment summaries and (2) a graph consisting of annotated relations between all comments in a dialogue; this is the gap that we aim to fill with our corpus creation work in Chapters 6 and 8.

Outside Argument Mining, narrowly defined, one corpus which is relevant to our work is the SENSEI annotated corpus (Barker et al., 2016). This corpus contains a number of online newspaper comment sections, many of which contain argument, together with summaries of comment clusters. However, no argumentative relation annotations are present in this corpus in its original form; in Chapter 8, we will describe the process by which we create an annotated dataset containing ASGs using SENSEI as a data source.

Online Resources One natural data resource to be considered for this task, given the fact that it was an inspiration for the graph structures used in this thesis, is kialo.com, a website which consists of a repository of user-edited argument trees. This website is structured as a database of debate topics. Each debate topic consists of a controversial proposition as a root node, with comments attached to the root either attacking or supporting it. These lower-level comments can then have comments which attack or support them, and so on. This structure is effectively identical to the one we propose. The main element it lacks from the point of view of our proposed task is that the propositions attached to each node are already summaries and do not have corresponding source comments.

Two key advantages of kialo.com are its large size (with over 24,000 debates at time of writing) and, due to the fact that decisions are made collectively by a team of enthusiast editors (in the style of Wikipedia) a degree of reliability that may be lacking in sources with fewer editors. Unfortunately, however, we cannot use it as a source due to copyright; on request, the website owners explicitly denied this permission.

Several other sources exist online which aim to collate common debate "points" across multiple topics. These include procon.org, debatepedia.org, and idebate.net (debatabase). Each of these three sites is similar, in that they are organised around controversial propositions, each of which has its own list of "pro" and "con" arguments. The disadvantage of such sites compared to kialo.com is clear; such sites provide only a very shallow argument tree of depth 1 (where all arguments attack or support the main topic). The one exception to this is idebate.net, whose structure includes counterpoints to the depth-1 points, resulting in a tree structure of depth 2. Because of this slightly deeper structure, we will use idebate as a data source in Chapter 6.

5.3 The ASG Parsing Task

In this section, we define the task of Argument Structure Graph Parsing. We first describe the target text types for the task (Section 5.3.1), before moving on to a formal description of the type of graphs that we aim to generate (Section 5.3.2), next, a description of a pipeline approach extracting ASGs into which we propose our task should fit (Section 5.3.4) and finally, a formal definition of the proposed task (Section 5.3.5).

5.3.1 Target Text Types for ASGP

Our definition of ASGP in this chapter will be mainly given in the context of its application to online comment threads. We have described the unique features of this text type in Section 1.3 of Chapter 1, and analysing this text type is a key motivation for the development of ASGP, due to the ubiquity and availability of this type of data. Futhermore, the two text resources that we develop in Chapters 6 and 8 are both based on online dialogue.

However, the ASGP task could equally well apply to spoken dialogues. Although we will talk about "comments sections", and "comments" throughout the rest of this chapter, the reader can take these techniques as equally applying to spoken dialogues, if we replace the words "comments section" and "comment" in the description below with "dialogue" and "utterance".

5.3.2 Argument Summary Graph Formalism

Before defining the task, we first describe the sort of output we are interested in generating. The argument graphing scheme that we choose is very simple, inspired by a simplified version of Freeman's argument graphing formalism (Freeman, 2011) and the graphs on the Kialo debate visualisation website (Kialo, 2024).

The ASG is tree structured, with the root of the tree being the main topic *M*. *M* is a controversial statement around which the argument centres; importantly, it must be a proposition rather than a simple noun phrase, e.g. "Britain should build more nuclear power plants", instead of just "nuclear power". We will explain our motivation for this constraint in Section 5.3.3.

Each of the non-root nodes in the tree has a unique label (e.g. "Summary 1"), and multiple attributes, of which the most important is the "summary", a short text that summarises one or more comments in the input. Examples of comments and their summaries from the Debatabase-ASG dataset (described in Chapter 6) are shown in Figure 1.1.

Additionally, each node may have an additional "source comments", attribute, an attribute representing the set of comments in the original comment thread to which the summary corresponds. A node may have multiple corresponding "source comments" when there are "redundant" comments in the input, i.e. when there are multiple comments stating an equivalent point. However, due to practical constraints on our dataset generation processes described in Chapters 6 and 8, in this thesis we have only explored datasets where each summary corresponds to a single source comment.

The argumentative relations between the nodes are labelled as "support" or "attack" — this represents the stance of the child node towards its parent

node.

The motivation for the use of a tree structure instead of a less constrained graph type is due to an assumption about the argumentative discussion: disagreements on a particular topic branch downwards to disagreements about a sub-topic that the parent argument depends on. Furthermore, using a less constrained graph type would conflict with our goal of producing a simple, human-readable summary of an argument. However, it is important to note that other authors have addressed the task of producing non-tree argument graphs, e.g. Morio et al. (2020).

5.3.3 The Main Topic (M)

As stated above, we restrict the main topic M to be a text that expresses a proposition, rather than simply a noun phrase, e.g. it should be something like "Britain should invest in constructing three additional nuclear power plants" rather than "nuclear power". This separates this work from some previous AM work which has aimed to classify arguments as "pro" or "con" a given "topic" that is stated as a noun phrase (e.g. Stab et al. 2018a).

In everyday conversation, informal debates are sometimes framed using only a noun phrase as the debate topic, e.g. between two sides who are "pro-nuclear" or "anti-nuclear". However, this characterisation is in reality meaningless; does a "pro-nuclear" person want to build a single power plant, or make the entire electrical grid nuclear (and in which country)? Furthermore, the same stance could be characterised as "pro-nuclear" or "anti-nuclear" in different contexts: for instance "build a single power station" could be taken as pro-nuclear among people who are extremely anti-nuclear, and as anti-nuclear among others.

One might argue that since our aim is to capture real-world debate, and since people may say things along the lines of “we are discussing nuclear power", that we should allow the possibility of a non-propositional debate topic. However, we would argue that any coherent debate is in fact implicitly about a proposition.

Real-world debates do exist which are incoherent, e.g. where one participant believes the debate is about proposition p and another believes it is about proposition q. In this case, we would lose something by assuming that a debate must be about a single proposition. However, we assume that these are rare enough that disallowing them in our scheme is not a severe limitation.

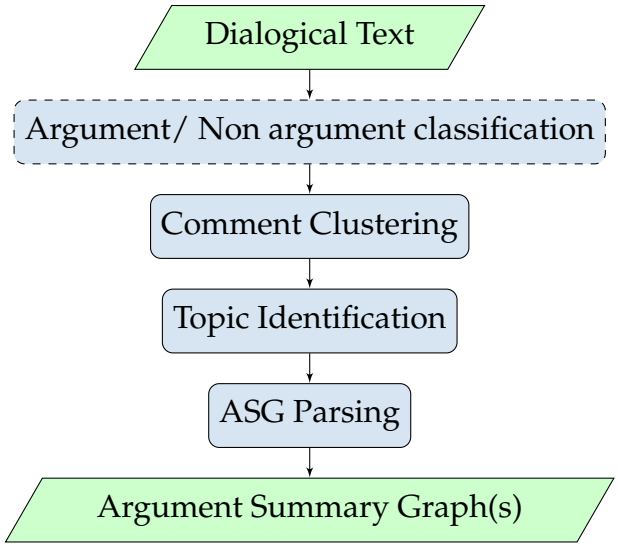


Figure 5.1: Our conception of the way in which ASG Parsing could be used within an Argument Summarization pipeline.

5.3.4 A Possible Pipeline For ASG Parsing

As we have defined ASGs, they are tree structured, with the argument branching down from a single controversial topic. However, online comments sections may actually contain debate about multiple topics, and some irrelevant

comments, as we have shown in Chapter 1, Section 1.3.

Due to these features, it may not be possible to directly generate an ASG from the text itself. We therefore propose that a pipeline approach be used to generate ASGs. The ASGP task, strictly defined, would only form the last stage in this pipeline.

In order to generate an ASG, therefore, we need a system that can cluster comments by argumentative topic, and can eliminate comments that are irrelevant. We propose that this can be achieved by a pipeline approach, which contains three modules in addition to an Argument Summary Parsing module. Such an approach is shown in Figure 5.1.

In this pipeline approach, we firstly use an argument/ non-argument classifier to remove all non-argumentative comments (such as asides and jokes). Next, a comment clustering component is used to separate the argumentative comments into different argumentative topics. Subsequently, the topic identification component identifies the "topic" or controversial proposition that each cluster is arguing about. Finally, ASGP is carried out on each topic cluster.

The dashed line around the argument/ non-argument classification component indicates that it may not be required, depending on the approach taken to ASGP. If we conceive of an ASGP module which is capable of removing irrelevant comments, then it may be possible to omit this component from the pipeline entirely. Furthermore, the ordering of steps in this pipeline is also tentative; for instance we could reverse the order of steps 1 and 2 and first cluster all comments by topic before deciding if they are argumentative or not. This is a question that future work should address empirically.

5.3.5 Task Definition

We now turn to formally defining the Argument Summary Graph Parsing task. We propose two different variants of the task, which differ in the input given to the model. These correspond to two different variants of the SENSEI-ASG dataset which we will develop in Chapter 8.

Variant 1: Comments only — in this variant, we do not include any reply structure in the input, but only a list of comments. We propose this variant because it is interesting, from a purely academic point of view, to test whether models are capable of generating argument structure without the benefit of reply-to information, which often closely corresponds with argument structure.

Variant 2: Comments with reply structure — in this variant, the reply structure from the original comment thread is included in the input. This is a realistic thing to include since reply structure exists on most online comments sections and fora. We therefore expect this variant to be a more useful task in a real-world setting.

Formally, we define the ASG parsing task as follows: for **variant 1** the input consists of a Main Claim M, the controversial proposition which is the main topic of an argument, and a set of comments $C = \{c_1, c_2, \ldots, c_n\}$, where each c_i expresses a stance toward a given main claim M or other comments. Note that C and M may be the result of a preprocessing pipeline run over an initial set of raw comments taken from a social media source, such as that depicted in Figure 5.1.

The goal is to construct a tree T rooted at M, and to produce a concise summary s_i for each comment $c_i \in C$. Each node in T is either the main

claim M or a summary s_i corresponding to a relevant comment c_i. Edges in T represent support or attack relations between summaries and M, or between summaries themselves (i.e., $s_j \rightarrow s_i$ indicates that s_j supports or attacks s_i).

Alternatively, in **variant 2**, we start out with an input tree U which is constructed from a set of comments C and a Main Topic M, and is rooted at M. Again, C and M may be the result of a preprocessing pipeline shown in Figure 5.1. In this variant, the input tree U is assumed to have been constructed in a prior step using the reply structure from the source comments. For each comment c_j, if its parent comment c_i within the original discussion is contained in C, it will be assigned as a child node of c_i. Otherwise, it will be made a child node of the main topic M. The required output for variant 2 is structurally identical to that of variant 1.

5.4 Evaluation

We have identified three separate aspects of ASGs that it is desirable to evaluate, and propose metrics to be used for each one. These are the following:

1. The quality of the textual summaries at each node
2. The correctness of the graph structure
3. The correctness of the support/attack labels

We consider various metrics for the evaluation of each of these three aspects, only some of which are used in later chapters of the thesis. These metrics are summarised in Table 5.1.

Aspect to Evaluate	Metrics
Quality of Summary Text	**ROUGE**, BERTScore, MoverScore
Graph Structure Evaluation	**Graph Edit Distance, Node-Position-F1**
Attack/ Support Label Evaluation	**Node-Stance-F1**

Table 5.1: A summary of the metrics described in this section. The metrics which we actually use in this thesis are shown in bold.

5.4.1 Quality of Summary Text

The first aspect we evaluate is the quality of the text, without reference to the graph structure. In this subsection we review two alternative ways of evaluating this: namely ROUGE and LLM-based metrics.

The general procedure that we have taken for evaluating summary text is to concatenate the text from every summary node within each respective ASG (gold standard and predicted) into a single string, and then run an evaluation function over this pair of strings.

This approach is only one possible choice; a second alternative would be to first make an alignment between all nodes in the output ASG and all nodes in the input, calculate the metric between all input-output pairs, and then take the average of these scores as the measure of summary quality.

We choose not to take the latter approach for two reasons: firstly, it is more computationally expensive to calculate metrics over many summary pairs, and secondly, if we align the nodes first, this will penalise those models which are poor at generating the output graph structure. We use other metrics (described below) to evaluate this, and we want our summary metrics, as much as possible, to only evaluate the text-quality aspect of model performance. We will now turn to describing the specific metrics which may be used to evaluate summaries.

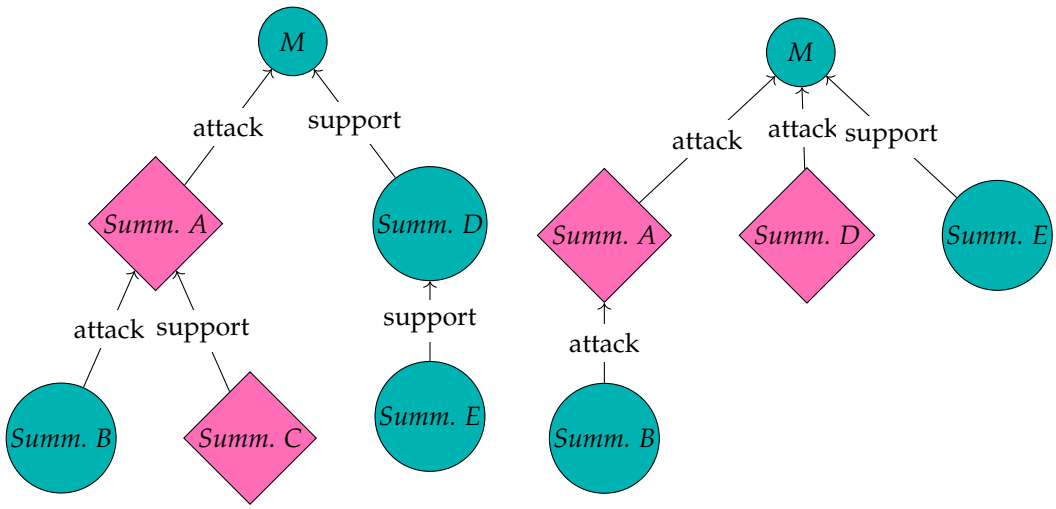
ROUGE (Lin, 2004) is a widely-used standard metric for measuring the quality of summaries. It works essentially by measuring the overlap of n-grams between one or more gold-standard references and the generated summary. It has advantages over other metrics due to the fact that it is conceptually extremely simple, it is well-established with multiple open-source implementations being available, and it has a low computational cost when compared to LLM-based metrics. As we will see in the section on LLM-based metrics below, it also seems to correlate with human judgement about as well as these more recently developed metrics do on summarisation tasks.

We use ROUGE in each of the chapters in which we carry out experiments on ASGP (Chapters 6 and 9). The implementation which we use is the py-rouge Python package (Antognini, 2020). We specifically use ROUGE-2 (bi-gram ROUGE) (Lin, 2004), which we have chosen due to it being a commonly-used variant which has been found to have a closer correlation with human judgement compared to other variants; additionally, we found that with only one set of reference summaries in our corpora, variants relying on longer sub-sequences (ROUGE-3 and above) have very small values. This could pose a problem due to a type of floor effect; if none of the summaries produced by the models we are testing share trigrams with the reference summary (or only one or two), it becomes harder to distinguish between model performance.

LLM-based Metrics It is important to note that a range of metrics are available for evaluating summaries. LLM-based metrics, which measure the distance between embeddings of two texts generated using a Large Language Model, have been growing in popularity in recent years; these include BERT-Score (Zhang et al., 2019) and MoverScore (Zhao et al., 2019). We might intuitively expect these metrics to have an advantage over lexical overlap based metrics such as ROUGE, due to the fact that LLM representations can capture the semantics of the input and abstract away from linguistic surface phenom-

ena to a degree; we may expect that they can therefore deal better with paraphrasing.

However, despite the fact that BERTScore has been found to correlate better with human judgements than previous state-of-the-art metrics in machine translation (Zhang et al., 2019), there is as of yet no evidence that it outperforms ROUGE in text summarisation. Empirical studies in this domain which have looked at the correlation of metrics with human annotation (Bhandari et al., 2020; Deutsch and Roth, 2021) have failed to find that LLM-based methods perform significantly better. In our work in this thesis, we have used ROUGE in evaluation due to its significantly lower cost (in terms of processing time) compared to the LLM-based metrics.



(a) Gold standard ASG

(b) Generated ASG

Figure 5.2: An illustration of a "gold-standard" ASG from a hypothetical test dataset, and a tree generated by a model exhibiting errors.

5.4.2 Graph Structure Evaluation

In this thesis, we make use of two different metrics to evaluate the correctness of the graph structure. The first is Graph Edit Distance (GED), a commonly used metric for finding the distance between two graphs. The second is Node Position F1, a simple metric that we propose for evaluating the accuracy of the position of tree nodes against a gold standard. The advantage of this metric against GED is that it normalises to 1 and so is more easily interpretable; the disadvantage is that it is only defined for trees, and so would not work with other types of Argument Graph.

Graph Edit Distance (Sanfeliu and Fu, 1983) is a metric similar to the Levenshtein distance (Levenshtein et al., 1966) but for graphs instead of strings. It is essentially a measure of how many steps are required to transform one graph into another. Formally, the graph edit distance between two graphs g_1 and g_2, $GED(g_1, g_2)$ can be defined as

$$GED(g_1, g_2) = \min_{(e_1,...,e_k) \in \mathcal{P}(g_1,g_2)} \sum_{i=1}^{k} c(e_i) \tag{5.1}$$

where $\mathcal{P}(g_1, g_2)$ denotes the set of edit paths transforming g_1 into g_2, $(e_1, ..., e_k)$ is a sequence of graph edit operations comprising an edit path. Possible edit operations include node insertions, deletions and substitutions, as well as edge insertions, deletions and substitutions. The function $c(e)$ is the cost of each graph edit operation e. In our case, we use the commonly employed default weightings for each edit operation, i.e. an insertion cost of 1, a deletion cost of 1, and a substitution cost of 2. We ignore edge labels entirely while scoring GED (i.e. we treat an "attack" edge as identical to a "support" edge), since we want to evaluate correctness of graph structure separately to correctness of edge labelling, which we look at in the next subsection.

In order to determine whether two nodes are "identical" for the purpose of calculating GED we use a simple node matching function, nodeMatch, which matches nodes based on their labels. Where label(v) is the function which returns the label of a given node (e.g. "Summary A" in figure 5.2, we define nodeMatch as follows:

$$\text{nodeMatch}(v_1, v_2) = \begin{cases} 1, & \text{if label}(v_1) = \text{label}(v_2) \\ 0, & \text{otherwise} \end{cases} \tag{5.2}$$

The generated graph in Figure 5.2, for example, would have a GED of 6, since it can be transformed into the gold-standard graph via 1 node deletion and 1 edge deletion (Summ. E and its connected edge) followed by 2 node insertions and 2 edge insertions (Summ. C and Summ. E and their respective edges).

Node Position F1 While Graph Edit Distance is a useful metric, the disadvantage for our purposes is that it is unnormalised, and therefore less useful when comparing performance between datasets. We therefore also use an alternative method of scoring tree structure that we will refer to as Node Position F1.

Node Position F1 scores graphs based on the position of each node relative to the root: nodes are considered to be positioned correctly if the path from the node to the root is identical in the predicted graph and the reference.

As shown in Figure 1.1, each node in the tree has a label, such as "Comment 1". We define a function $lab(v, T)$ which returns the label of a given node v in a tree T. For each node v, we can then also define the root-to-node path, denoted by $P(v, T)$. We define $P(v, T)$, where $v_0 = v$, $v_k = \text{root}(T)$, and for each i such that $0 \leq i < k$, v_{i+1} is the parent of v_i in T:

$$P(v, T) = (\text{lab}(v_0), \text{lab}(v_1), \ldots, \text{lab}(v_k)) \tag{5.3}$$

The set of all node-to-root paths $\mathcal{P}(T)$ in T is then the collection of all such sequences for each node $v \in V$:

$$\mathcal{P}(T) = \{P(v, T) \mid v \in T\} \tag{5.4}$$

We can then define the sets of all node-to-root paths in the ground truth tree and the generated tree respectively:

$$P_{true} = \mathcal{P}(T_{true}) \tag{5.5}$$

$$P_{pred} = \mathcal{P}(T_{pred}) \tag{5.6}$$

We can then define True Positives (TP), False Positives (FP) and False Negatives as below:

$$TP = |P_{\text{true}} \cap P_{\text{pred}}| \tag{5.7}$$

$$FP = |P_{\text{pred}} - P_{\text{true}}| \tag{5.8}$$

$$FN = |P_{\text{true}} - P_{\text{pred}}| \tag{5.9}$$

Node Position F1 can then be calculated as follows, based on the well-known F1-score: (van Rijsbergen, 1979, p. 114).

$$F1 = 2 \cdot \frac{TP}{2TP + FP + FN} \tag{5.10}$$

5.4.3 Attack/ Support Label Evaluation

One option for evaluating the quality of the attack/ support labels would be to directly evaluate their correctness using the F1 score, as is sometimes done when evaluating Argument Structure Parsing (see Section 7.5). However, a problem with this is that we have no way of evaluating edge labels on edges which differ from the gold standard. This can occur either when the structure generated is incorrect, or where the model has generated an alternative structure which is equally as valid as the ground truth. For example, if we consider three summary nodes, *M* (the main topic), *A* and *B* where the main topic *M* is "We should invest in nuclear power", summary *A* is "Nuclear energy is clean" and summary *B* is "Nuclear power stations generate dangerous waste", then it is ambiguous whether node *B* should be taken to attack node *A* or the topic node. Figure 5.3 illustrates the two possible structures.

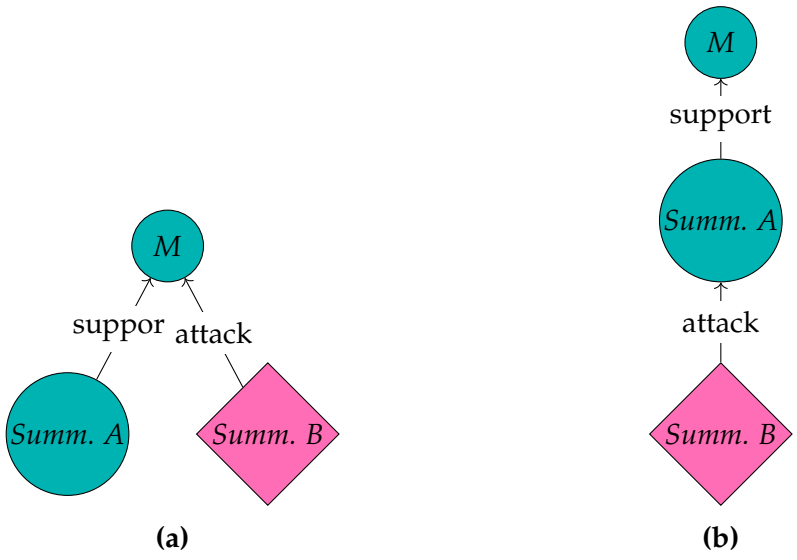


Figure 5.3: An illustration of a "gold-standard" ASG from a hypothetical test dataset, and a tree generated by a model exhibiting errors.

The alternative that we propose is called **Node Stance F1**. The idea is that instead of evaluating attack/support relations for the edge labels, we assign

pseudo-stances to nodes based on the edge relations, and then calculate an F1 score based on these stances. There are two possible stances, positive (+) and negative (-).

Firstly, we assign the stance of the main topic node (M) as positive:

$$\text{stance}(M) = + \tag{5.11}$$

We then iteratively assign a stance to the remaining nodes in the tree, top-down. For each child node v of a parent node u we assign that node to have the same stance as u if it has a "support" relation with its parent, or the negation of the stance of u if it has an "attack" relation with its parent. Formally:

$$\text{stance}(v) = \begin{cases} \text{stance}(u), & \text{if } (u,v) = \text{ support} \\ -\text{stance}(u), & \text{if } (u,v) = \text{ attack} \end{cases} \tag{5.12}$$

When we assign stances in this way, it means that nodes whose position in the graph structure is ambiguous, such as node B in Figure 5.3, will be evaluated as correct as long as the stance towards the root is accurate.

Once we have assigned stances, we can then calculate the F1 score for both "positive" and "negative" nodes against the ground truth, using the standard F1 score equation:

$$F1 = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{5.13}$$

F1 score is calculated for each of the two classes, attack and support. We finally calculate the overall macro-averaged Node Stance F1 score as the mean of these two F1 values.

The disadvantage of this metric is that in some cases, the assumptions we have made about the transitivity of support/attack relations do not hold. For example, in the chain:

$$A \xrightarrow{\text{attacks}} B \xrightarrow{\text{attacks}} C \tag{5.14}$$

it is not necessarily the case that the stance of A towards C is positive. However, in the case of debates which are around polarised issues with two possible stances, which constitute much of the data that we will look at in the following chapters, we consider this a reasonable simplifying assumption.

It should be noted that the algorithm described above is similar to a number of labelling-based algorithms used in the field of Abstract Argumentation Theory, also known as Dung Semantics (Dung, 1995). These algorithms are used to find “acceptable sets", that is, conflict-free sets arguments within an argument graph (see Toni 2020 for an overview of this field). However, our algorithm is significantly simpler since we only consider trees instead of general graphs.

5.5 Applications of ASGs and Post-Processing

In this section, we will consider how ASGs could be further processed after their generation. While they may be useful by themselves as an aid for understanding an argumentative discussion, there are at least two ways in which they could be processed further:

1. By aggregating multiple ASGs which address the same topics, for a broader view of debate across texts or platforms
2. By using them as a basis for generating a text-based summary

We will address each of these in turn.

5.5.1 Aggregating ASGs

As described so far, ASGs have been viewed as a tool for understanding a single text. However, for other applications it may make sense to aggregate multiple ASGs across texts—for instance, to understand a "hot topic" being discussed across several news websites. One way to do this is to combine multiple already-generated ASGs into a single graph.

The question of how to combine ASGs is non-trivial, as there are several alternative methods that may produce different results.

Assume that we have two ASGs to combine, T and U, and a similarity function f that calculates similarity based, for example, on the summary text alone. One possible approach is to iterate over all pairs of summary nodes and "merge" any two nodes whose similarity exceeds a given threshold. While this is a valid approach, the resulting structure would not necessarily be a tree, but rather a directed graph, and may contain cycles.

Alternatively, we can design aggregation algorithms that preserve tree structure. One way to do this is to iterate over T in a breadth-first manner (excluding the root). For each node t in T, we compare it using f to every node in U, also in breadth-first order (again excluding the root). If a node u matches t, we prune u and its descendants from U and merge the resulting subtree into T at t. After completing the iteration, we merge the roots of T and U. A disadvantage of this approach is that the final tree may still contain duplicate nodes.

5.5.2 Textual Summary Generation

Another possible application of ASGs is the generation of textual summaries. Barker and Gaizauskas (2016) propose several simple algorithms for the generation of textual summaries from an argument graph, whose structure is not too dissimilar to an ASG. Algorithm 5.1 is a reproduction of one of the algorithms proposed in this paper. In their graph representation, nodes are divided into "issues", "viewpoints" and "rationales". The "issues" are similar to the "main topic" in our representation, and the "viewpoints" and "rationales" correspond roughly to our summary nodes.

Their algorithm first sorts the "viewpoints" on an issue according to an importance function `fS`. It then iterates over each of the sorted viewpoints, first appending the viewpoint to the Summary, and then sorting the rationales which are the children of the viewpoint by a separate importance function for rationales, `fE`.

It is easy to conceive of a similar algorithm being developed for our ASG representation. The key direction of research for carrying this out would be establishing a suitable importance function for ranking the importance of summary nodes, analogous to Barker and Gaizauskas's `fS` and `fE` functions. Some intuitively plausible features which could be explored for developing such a function include the subtree size of a node v (that is, the total number of descendants of v), and the semantic similarity of v to the main topic M.

Additionally, we might ask whether an iterative approach similar to the one proposed by Barker and Gaizauskas is ideal, or whether the task could be approached end-to-end, taking advantage of neural techniques. The main bottleneck for developing a model to do this, and also for evaluating approaches to this task, is the lack of a corpus containing ASGs paired with free-text summaries. Any further work on this task should first address this lack of data.

Algorithm 5.1: *Algorirthm 1 from Barker and Gaizauskas (2016), which is referred to as a "simple single-issue summariser". The algorithm selects components from an argument graph and combines them into a textual summary. This is a general scheme for how such an algorithm could be developed, so implementation details such as the arguments of various functions are undefined.*

```
"""
Variables:
-G: An argument graph
-I: An ``issue'' node in G
-ThresholdE: A threshold for filtering evidence nodes by the measure
    of importance calculated by fE

Functions:
-get_viewpoints(G, I): returns a list containing all viewpoints
    towards issue I in graph G
-fS(.,.): A function for computing a measure of the importance of a
    Summary node. Arguments are left undefined.
-fE(.,.): A function for computing a measure of the importance of an
    Evidence node. Arguments are left undefined.
"""

def summarize_argument_graph(G, I, fS, fE, ThresholdE):
    Summary = []
    Summary.append(I)
    S = get_viewpoints(G, I)
    S.sort(key=fS)
    for s in S:
        Summary.append(s)
        Rs = get_rationales(G, s)
        Rs.sort(key=fE)
        for rs in Rs:
            if rs > ThresholdE:
                Summary.append(rs)
    return Summary
```

5.6 Conclusion

In this chapter, we have described our proposed task, ASG Parsing, which we will go on to investigate in the remaining thesis chapters. We have firstly situated this task in the context of previous research, and described our motivations for proposing the task. Secondly, we have defined the ASGP task

formally, and proposed a general pipeline approach within which it could be used for ASG extraction. Thirdly, we have reviewed three classes of metrics that can be used to evaluate different aspects of the ASGP task output, and explained why we have chosen the metrics used in later chapters of this thesis. Finally, we have explained some possible downstream applications of ASGs.

Chapter 6

Preliminary Experiments on Argument Summary Graph Parsing in Online Debate

6.1 Introduction

In Chapter 5, we introduced the task of Argument Summary Graph Parsing (ASGP), in which a system must be designed to create a graphical summary of an argumentative dialogue. In this chapter, we turn to a first attempt at carrying out that task.

We describe the creation of a dataset designed to support the training and testing of algorithms that address the ASGP task, as well as the experiments that we have carried out with the dataset. These experiments involve comparing several different approaches to ASGP. The main contributions of this chapter are the following:

1. We release a dataset for developing algorithms that address the ASGP task. Our dataset is derived from 590 debates available on the *Debatabase* website[10].

[10] idebate.net

2. We compare the effectiveness of a multi-stage LLM pipeline to a single end-to-end model using the TANL (Translation between Augmented Natural Languages) framework (Paolini et al., 2021) for the task of ASGP. We show that the TANL models substantially outperform the pipeline approach across different LLM models and metrics.

3. We evaluate the performance of various language models on the ASGP task, as well as In-Context-Learning (ICL) vs. fine-tuning strategies.

4. We compare two different tree depths for the ASG output: depth-1 and depth-2. Depth-1 trees are those where every non-root node links to the root, while depth-2 trees also contain nodes that link to the children of the root. We show that performance for all models degrades with even this slight increase in tree depth, highlighting a limitation of the approaches explored here.

6.2 Background

Since this is a novel task and dataset, we cannot directly compare our approach to any previous approaches. However, this task has a number of similarities with certain tasks in Argument Mining, such as Argument Structure Parsing, so it is worth considering some previous techniques used in ASP for application to this task. In ASP, we also have to generate, from a text, a graph structure with labelled edges corresponding to argumentative relations. The node types in the output, however, also differ between the two tasks: in ASP the output nodes are Argumentative Discourse Units (see Chapter 2, Section 2.3.3 for a definition), which are text spans that appear in the original text. In ASGP, in contrast, nodes in the output are summaries of a comment or comments in the original discussion, and therefore do not appear verbatim in the input text (see Chapter 5 for a full task description of ASGP).

A large variety of models have been applied to the ASP task, even if we only consider those that have been applied on the Argument-Annotated Essays Corpus (see Chapter 7, Section 7.2 for details). However, in this chapter we consider only two techniques, both due to their relative simplicity of implementation for the first attempt at this task, and to their strong performance in ASP.

6.2.1 MST Parsing

One of the approaches from ASP which we consider here is called MST (Maximum Spanning Tree) parsing. This is an approach that was originally used within NLP for the task of parsing text into a syntactic dependency tree representation (McDonald et al., 2005).

In this technique, we begin with a set of the "node" elements from which we have to construct a graph in the target task (in syntactic parsing, these are words, and in ASP, they are Argumentative Discourse Units). The first step is then to build a "weight matrix" which represents the probability of an edge existing between all possible (ordered) pairs of these nodes. Any of a number of machine learning approaches can be used in principle to generate this weight matrix. Within ASP, this matrix has been generated using LLMs in Morio et al. (2020). LLM classifier models were used to predict the pseudo-probability of an edge between a pair of ADUs existing, by using the two ADU text spans as input to the model.

Once a weight matrix is constructed, a Maximum Spanning Tree (MST) algorithm is then run over the matrix. MST algorithms are a class of algorithms which return the tree that contains the combination of edges which sums to the largest value, which in this case is equivalent to finding the most "probable" tree, given the predicted edge weights.

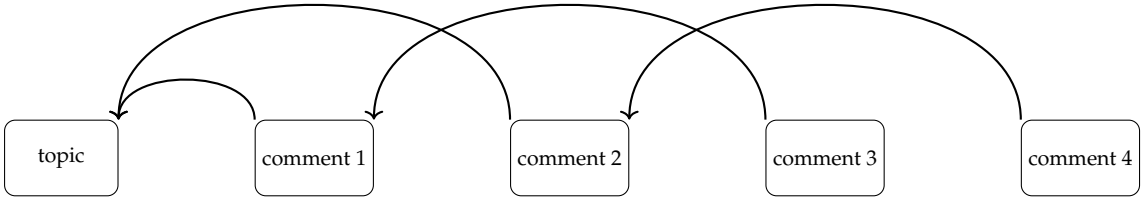


Figure 6.1: Illustration of a Non-projective comment structure tree. The arrows link a "child" comment to the "parent" comment that it responds to.

One way in which MST algorithms vary is in whether or not they make assumptions about the "projectivity" of the trees they generate. This is a concept which originates from the field of syntactic dependency parsing, in which trees are defined as "projective" or "non-projective". A "projective" dependency tree is a tree such that, when the tree nodes are placed in linear order (i.e. the order in which they appear in the original text), it is possible to draw all of the tree edges above the nodes without at least one line crossing another, and a non-projective tree is one for which this is impossible. Nivre (2008, p. 517) provides a formal definition of the concept.

One MST algorithm commonly used in parsing is the Chu-Liu Edmonds algorithm (Chu, 1965; Edmonds et al., 1967), which is useful because it makes no assumptions about the projectivity of trees, unlike some other algorithms which enforce projectivity. Argument Mining researchers working on ASP, such as Morio et al. (2022), Peldszus and Stede (2015b) and Huang et al. (2023), have also used the CLE since Argument Graphs need not be projective.

It appears that for our task, too, non-projective trees are possible. Figure 6.1 illustrates an example of a non-projective tree structure that may appear in a comment section. If we assume that the ASG structure corresponds to the reply structure in the comment tree — which we do in this dataset — then the ASG that results from correctly parsing this tree will also be non-projective. Therefore, we will also use the CLE algorithm for this task.

6.2.2 Text-to-text Approaches

Another possible technique is to carry out the entire task end-to-end with a single LLM, framing the problem as a text-to-text "translation" task. We will refer to this method here as the TANL (Translation between Augmented Natural Languages) approach, following the terminology of Kawarada et al. (2024), who themselves took it from Paolini et al. (2021). This is a general framework in which an LLM is trained to infer latent structured information contained in an input string, by learning to "translate" it to an output string which encodes this structured information in some sort of formal language.

Kawarada et al. (2024) apply TANL to Argument Structure Parsing. They fine-tune an LLM (Flan-t5-XXL, Chung et al. 2022), using the entire target text (in their case, an argumentative essay) as input, and a copy of that essay, augmented with a markup-style language, as the output. Example 6.1 illustrates what the input and output looks like for an essay fragment.

Example 6.1: Example of the TANL formatting used in Kawarada et al. (2024)

Input: As far as I am concerned, we should do our parts to do what ever it takes to protect old buildings, letting next generations still have a chance to look at them. Here are some reasons and examples to illustrate my viewpoint. Historic buildings have intrinsic values, because they are significant symbols of a city.

Output:
As far as I am concerned, [we should do our parts to do what ever it takes to protect old buildings, letting next generations still have a chance to look at them MajorClaim]. Here are some reasons and examples to illustrate my viewpoint. [Historic buildings have intrinsic values Claim for], because [they are significant symbols of a city Premise supports = Historic buildings have intrinsic values].

This example demonstrates how TANL works — each ADU text span within the output (i.e. the spans within square brackets) has been annotated, using a markup-style language, with information which represents an argument graph. In this case, that includes the ADU type, e.g. "MajorClaim", and also, where applicable, the parent ADU and the relation to the parent, e.g. "supports = Historic buildings have intrinsic values".

In this chapter, we adapt the TANL approach to ASGP. We experiment with training LLMs to convert an input string representing an entire comment section to a string which represents an ASG. The formatting that we propose differs from that used by Kawarada et al. (2024), and will be explained in Section 6.4.3.

6.3 Research Questions

We investigate three research questions in this chapter. These are motivated by the need to investigate our novel task, as well as the approaches that have been taken to ASGP mentioned in the Background section above. We enumerate these research questions below:

RQ 6.1 — Which approach works best for ASGP: a pipeline approach or an end-to-end approach? As we have seen above, the task of ASP has been addressed both using MST parsing approaches, and the end-to-end text-to-text approach used by Kawarada et al. (2024). We propose two approaches for ASGP in this chapter that are based on those two task framings, and we carry out experiments with these using a range of LLMs in order to compare their performance.

Firstly, we propose a three-stage pipeline approach for ASGP consisting

of three modules: (1) a summarisation module, (2) a stance-detection module, and (3) an MST parser which uses the stance detection output to create a graph.

The second approach, based on that of Kawarada et al. (2024), frames the entire task as a text-to-text task, in which an LLM must concert the input text into a textual representation of the argument graph. Each of these approaches is described in greater detail in Section 6.6

RQ 6.2 — How does increasing the tree depth of ASGs affect task performance? As will be explained in Section 6.4, we are able to generate tree-structured Argument Summary Graphs of up to depth-2 from the *Debatabase* dataset. We generate two parallel datasets of depth-1 and depth-2 ASGs, depth-1 ASGs being trees in which all nodes are connected to the root, and depth-2 ASGs being trees in which the longest path from the root to any leaf consists of two edges.

There is a strand of research on transformer models that suggests that suggests that these models may struggle to process deeper hierarchical structures (Murty et al., 2023). We wish to investigate if this is also that case for ASGs, and if so, to what extent. Additionally, we would like to investigate which of the two task approaches (end-to-end or pipeline) is better at dealing with deeper structures.

RQ 6.3 — How effective is using In-Context-Learning for ASGP, compared to fine-tuning? As we have explained in Chapter 2, Section 2.6.2, there are several different alternative "learning paradigms" used in NLP to take advantage of pretrained LLMs. Previous work in ASP (e.g. Morio et al. 2022; Kawarada et al. 2024) and elsewhere in argument mining has largely used the fine-tuning paradigm, in which a pretrained model is subsequently trained on task-specific data.

However, one alternative paradigm is "In-Context-Learning" (explained in more detail in Section 6.6.5). In this paradigm, no additional task-specific training is carried out, but instead the model is simply prompted at inference time with multiple examples of input-output pairs from the target task. For some tasks and models, the task can be inferred from these input-output pairs, allowing the model to solve the task for a target example given in the same prompt. This is an approach which is overall much less computationally costly than fine-tuning, and therefore merits investigation into how well its performance compares to the fine-tuning approach.

RQ 6.4 — How well do different LLMs perform on this task? Finally, we compare the performance of various LLM base models; namely, those shown in Table 6.2.

6.4 Dataset

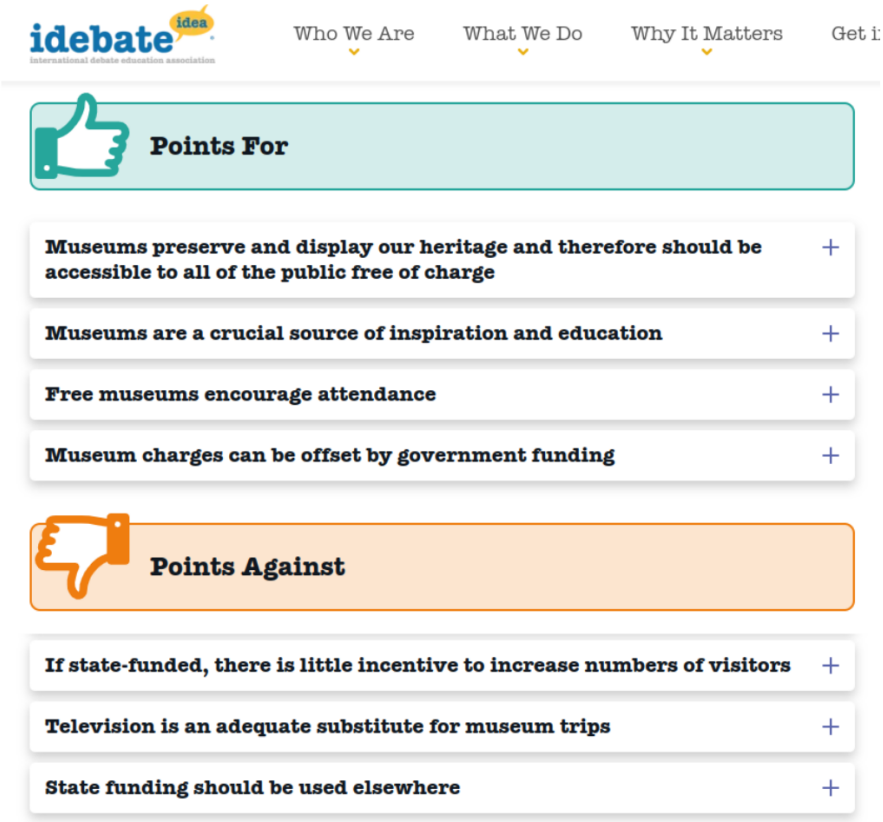
In this section, we describe the new resource that we have produced, Debatabase-ASG, which we have released publicly.[11] In this section, we describe the new dataset that we introduce; firstly describing the data source (*Debatabase*), secondly, the creation process, and finally, the statistics of the final corpus.

6.4.1 The Data Source

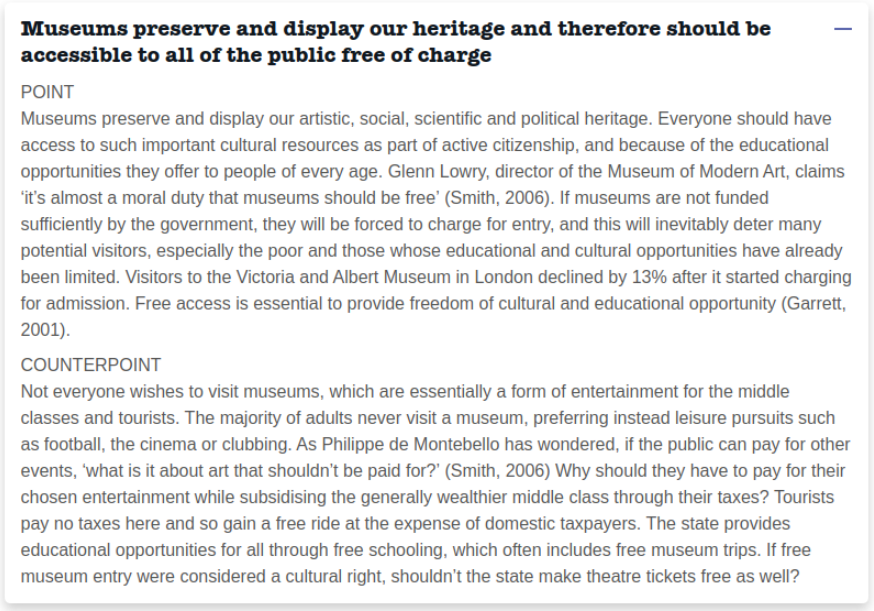
We sourced our dataset from the *Debatabase* website. [12] *Debatabase* is a manually curated collection of pros and cons on a range of controversial topics across

[11] github.com/acidrobin/ASGParsing

[12] Debatabase can be found at idebate.net/resources/debatabase. Idebate allows reproduction of materials on its website for noncommercial academic purposes, provided that its copyright is acknowledged (https://idebate.net/terms-and-conditions)



(a) "Points For" and "Points Against" section headings



(b) Each heading can be expanded to reveal a "point" and a "counterpoint"

Figure 6.2: Screenshots from the idebate website. Each "point" summary (shown in subfigure a) can be expanded to reveal a lengthier text explaining that point and its counterargument (shown in subfigure b).

multiple categories such as politics and culture. The topics are phrased as a conventional motion of the type that would appear in a competitive debate, for example "This house would make all museums free of charge", or "This house supports the idea of participatory democracy".

This data has been used in previous works, notably the args.me corpus (Ajjour et al., 2019). We cannot use data in that format for our purposes as it has been separated into sentence pairs for training stance classification models; we are instead interested in summarising whole discussions.

The data is in some respects not ideal due to its artificiality; it does not contain spontaneous online comment, but rather carefully curated and well-sourced arguments. Despite this, we have chosen to use it due to a lack of other sources of data amenable to the ASGP task. [13]

To produce an ASGP dataset, we require: (1) a set of comments, (2) a set of summaries of those comments, and (3) a set of directed attack/ support relations between those summaries, which will allow us to build those summaries. The *Debatabase* dataset contains elements which we can use for each of these. More specifically, we use the "point headings" as summaries, and the hierarchical tree structure of points and counterpoints (both shown in Figure 6.2). We will explain, in more detail, how these elements were converted into ASGs in the following subsection.

[13] It should be noted that we do not scrape data from kialo.com, arguably an even better source of summary graphs, due to copyright issues; permission to use for research was requested and explicitly denied by the copyright holders.

Main topic: This House would make all museums free of charge
Comment 1: Museums preserve and display our artistic, social, scientific and political heritage. Everyone should have access to such important cultural resources as part of active citizenship, and because of the educational ...
Comment 2: The majority of adults never visit a museum, preferring instead leisure pursuits such as football, the cinema or clubbing. As Philippe de Montebello has wondered...
Comment 3: Museums are a crucial source of inspiration and education for our increasingly important creative industries (e.g. art, design, fashion, and architecture)...
Comment 4: If museums are to be funded entirely out of public money, the pressure on any government's budget from the demands of hospitals, schools, pensions, etc. ...

(a) Input Text

Comment 1 (**supports** **main topic**): Museums preserve and display our heritage and therefore should be accessible to all of the public free of charge.
Comment 2 (**attacks** **Comment 1**): Not everyone wishes to visit museums, which are essentially a form of entertainment for the middle classes and tourists.
Comment 3 (**supports** **main topic**): Museums are a crucial source of inspiration and education.
Comment 4 (**attacks** **Comment 3**): Such potential and economic benefits are dubious and rely upon access to collections that are excellent in their content and in the way in which they are conserved and displayed.

(b) Output text

Figure 6.3: Example input comment thread and output summary graph from our dataset.

Main topic: This House would make all museums free of charge
Comment 1 : Museums preserve and display our heritage and therefore should be accessible to all of the public free of charge.
Comment 2: Not everyone wishes to visit museums, which are essentially a form of entertainment for the middle classes and tourists.
Comment 3: Museums are a crucial source of inspiration and education.
Comment 4: Such potential and economic benefits are dubious and rely upon access to collections that are excellent in their content and in the way in which they are conserved and displayed.

Figure 6.4: Intermediate text representation used in our two-prompt ICL method

6.4.2 Dataset Creation Process

We used a web-scraping script to process the entries on the website. Our script first filtered out invalid entries, i.e. entries which contained only a main topic and a debate description, but no points or counterpoints. Subsequently, the script processed the information inside the entry to produce both a "comment thread" and an ASG. Since we are interested in investigating how graph depth affects the difficulty of generating ASGs, we in fact produced two comment threads, and two corresponding ASGs, for what we call the "depth-1" and "depth-2" settings. We describe how we generated each in turn below.

Algorithm 6.1: *Dataset creation (depth-1)*

1. Create the comment thread:
   (a) Extract the text of every "point" in the entry.
   (b) Exclude all "counterpoints".
   (c) Randomly shuffle the extracted point texts.
   (d) Concatenate the shuffled texts into a single comment thread.
2. Create the ASG:
   (a) Use the main debate topic as the root node.
   (b) For each extracted point:
      i. Create a summary node using the corresponding "point for" or "point against" heading.
      ii. Attach this summary node as a child of the root.

Depth-1 setting: For the depth-1 setting, we generated the comment thread by taking every "point" text from Figure 6.2, and concatenating them in a random order (but not the "counterpoints). To generate the corresponding ASG, we take the main topic of the debate as the tree root, and add each "point for" and "point against" heading as a summary node. Algorithm 6.1 explains this process more fully.

Algorithm 6.2: *Dataset creation (depth-2)*

1. Create the comment thread:
   (a) Collect all "points" and "counterpoints" from the entry.
   (b) Randomly sample six items in total.
   (c) During sampling:
      i. If a "counterpoint" is selected, also select the corresponding "point".
   (d) Concatenate the selected texts into a single comment thread.
2. Create the ASG:
   (a) Use the main debate topic as the root node.
   (b) For each sampled "point":
      i. Create a depth-1 summary node using the point heading.
      ii. Attach this node as a child of the root.
   (c) For each sampled "counterpoint":
      i. Create a depth-2 summary node.
      ii. Use the first sentence of the counterpoint text as the node summary.
      iii. Attach this node as a child of the corresponding depth-1 point node.

Depth-2 setting: For the depth-2 setting, we included both the "points" and "counterpoints" from Figure 6.2. Our process for creating the comment threads and ASGs is explained fully in Algorithm 6.2. In order to ensure that the size of the comment thread (in terms of text length) did not expand beyond the context length of the models that we used in experiments, we randomly sampled six "points"/ "counterpoints" in total (In our sampling process, we ensured that if we sampled any "counterpoint", we also sampled the corresponding "point", so that the resulting ASG would have a valid tree structure).

As depth-1 summary nodes for the ASG, we used the headings corresponding to the randomly selected "points". Depth-2 summary nodes correspond to the "counterpoints". Since the "counterpoints" do not have corresponding subheadings on the website, we used the first sentence of the counterpoint text for each depth-2 node summary.

6.4.3 Statistics and Formatting of the Final Dataset

Table 6.1 shows statistics of the final dataset that we produce.

We store the data in a string format that is amenable to a TANL task framework. An example of this format is shown in Figure 6.3. The input is a single string consisting of lines concatenated by a "newline" symbol. The format of each line is `"{comment_id}: {comment_text}"`, where the `comment_id` can either be "Main Topic" or a comment number, e.g. "Comment 1".

The output is a concatenated list of summaries, with every line corresponding to one of the comments in the input. Each output line has the format `"{comment_id}: ({stance} {parent_id}): {summary text}"`, where "stance" is the stance towards the parent comment, and "parent number" is the number of the parent comment.

	Depth-1 trees	Depth-2 trees
N. comment threads	590	590
Avg. comment thread length (whole word tokens)	1365	1054
Avg. length of Gold standard ASG text representations (whole word tokens)	105	107
Avg. comments per thread	7	6

Table 6.1: Statistics of the data collected from *Debatabase*. Depth-1 and Depth-2 trees contain comments on the same topics, but with different tree depths (in Depth-1 trees every node must attack/support the root, in Depth-2 trees, nodes there is also always at least one node that is a child of a level-1 node).

6.5 Evaluation

We evaluate three aspects of the quality of the graph output, as explained in Chapter 5, Section 5.4. These are ROUGE, GED, and Node-Stance-F1. We use GED rather than Node-Position-F1 to evaluate the structure, since we had not developed the latter metric at the time of carrying out these experiments.

When evaluating the end-to-end models, we additionally encounter the problem of post-processing the textual output. As we will explain in more detail in the Experiments section (6.6), the end-to-end models are fine-tuned or prompted to produce an output in the format of Figure 6.3(b). The issue that we face is that LLMs can produce free-text output and are therefore not constrained to generate a valid graph in this format. For this reason, we used a simple post-processing algorithm that converted the output text into an unambiguous graph representation.

This algorithm works by using a regular expression which attempts to match each line with the formatting in 6.3(b). If the regex matches, then we check if the proposed new node's parent is already in the graph. If it is, then the node is added as a child of the parent; if not, then the node is simply not added to the graph.

Note that this algorithm will only work for post-processing tree structures in which lower-level nodes cannot precede higher-level nodes linearly (e.g. if Comment 4 is a child of Comment 3, it must appear *after* Comment 4 in the LLM output). This is true for the Debatabase-ASG dataset.

6.6 Experiments

This section contains five subsections. Firstly, we describe the baseline used (Section 6.6.1. Next, in Section 6.6.2 we describe the Large Language Models that we used, and the process by which we fine-tuned them, which remained constant for across approaches. We then compare provide details of the two approaches — pipeline and end-to-end — in Sections 6.6.3 and 6.6.4 respectively. Finally, in Section 6.6.5 we outline our experiments on In-Context-Learning, a technique which involves prompting the models without fine-tuning.

Table 6.2 gives an overview of the experiments that we carried out, which we will describe with more detail in the subsequent sections. As the table illustrates, we used two main approaches, "Pipeline" and "End-to-End".

"Pipeline" approaches used a pipeline of two language models, trained for two separate subtasks. In each case, the same base model was used for both subtasks; i.e. (both in each case of the same base model, for example a pair of Longformer models or a pair of Llama-2 models), fine-tuned for two

	Pipeline	End-to-end	
	Fine-tuning	**Fine-tuning**	**In-Context Learning**
Longformer	✓	✓	
Phi-2	✓	✓	
Llama-2	✓	✓	✓
GPT-3.5	✓	✓	✓

Table 6.2: Overview of the Experiments carried out in this chapter, sorted by task framework and LLM type

separate tasks: one for summarisation and the other for stance detection (it is important to note that the stance-detection model is used to carry out two "subtasks" — relation detection and stance detection — simultaneously; this will be explained in Section 6.6.3).

The end-to-end models, in contrast, treated the whole task end-to-end as a single text-to-text translation task. The end-to-end column is additionally broken up into the two separate approaches we used in this setting: fine-tuning and in-context learning.

6.6.1 Baseline

For the baseline, we treated the task as three separate subtasks: relation classification, stance detection, and summarisation, and used a simple baseline for each. For the relation classification component, we made all non-root nodes children of the root node (as this is the most common parent node in the dataset). For the stance detection component, we labelled all relations *attack*, since this is the majority class. Finally, for the summarisation component, we used the first sentence of each comment as its summary; this is a common baseline in the field of text summarisation.

6.6.2 LLMs and Fine Tuning

In this section, we describe the Large Language Models used in our experiments and the fine-tuning process. The inference procedures and data formatting differed between the two framings of the task (pipeline and end-to-end). However, the models used, as well as other details such as the hardware and the hyperparameters trained by grid search remained the same across both approaches; we summarise these details here.

Model	**Base model**	**Params**
Longformer	allenai/led-base-16384	150M
Phi-2	microsoft/phi-2	2.7B
Llama-2-7b	meta-llama/Llama-2-7b	7.9B
GPT-3.5	gpt-3.5-turbo-1106	175B

Table 6.3: Models Tested

The models tested are shown in Table 6.3. The same hyperparameters and models were used in both the TANL approach and in the two stage pipeline approach.

One constraint we had when choosing models to evaluate was requiring models that can process a long sequence length (using the Llama-2 tokenizer, the mean average sequence length for this task in the TANL setting was around 2500 tokens, and around 10% of samples had over 4000). Hence, we selected the Longformer model as the most appropriate "small" LLM. We then selected three additional recently-released LLMs in order to benchmark the performance of models with a range of sizes (in number of parameters), namely Phi-2 (Javaheripi et al., 2023), Llama-2 (Touvron et al., 2023a) and GPT-3.5 (Brown et al., 2020).

When fine-tuning, we use an 8:1:1 train-validation-test data split ratio. The splits are made by topic rather than by data point, such that identical topics do not appear across the three splits.

For GPT-3.5, one epoch of fine-tuning was carried out with the OpenAI API using the default settings; we did not carry out a parameter grid search due to cost constraints. We fine-tuned the other models (Longformer, Phi-2 and Llama-2-7b) on an HPC cluster using two NVIDIA HGX H100 GPUs. We used the AdamW optimizer (Loshchilov and Hutter, 2017). For each of these models, we carried out a grid search over the parameters shown in Table 6.4. The other hyperparameters which we did not tune are shown in Appendix A.3.

Hyperparameter	Values Tested
Learning Rate	{1e−3, 1e−4, 1e−5}
Weight Decay	{0.001, 0.0}
LORA Dropout (QLORA Models Only)	{0.1, 0.5}

Table 6.4: Hyperparameters tested by Grid Search for all models

For Phi-2 and Llama-2-7b, we lacked the computing resources to fine-tune the entire model. We hence used the QLoRA (Quantized Low-Rank Adapters) technique (Dettmers et al., 2023). This is a widely used Parameter Efficient Fine Tuning (PEFT) method, which allows for training large models using fewer computational resources (we have explained this in more detail in Section 2.6.3).

6.6.3 LLM Pipeline Approach

Our first approach, the pipeline approach, has two LLM components: a summarisation model and a stance detection model. In each pipeline experiment, we use the same model type for both components (i.e. a pair of Longformer models, a pair of Phi-2 models, and a pair of Llama-2 models). We use a different pipeline approach in the depth-1 and depth-2 settings. We first describe the approach we take for the depth-2 setting, and then depth-1.

Depth-2 Setting: In the depth-2 setting, our pipeline approach has three steps. **Step 1** is to use the summarisation model to generate a textual summary of each individual comment. **Step 2** involves running the stance detection model over pairs of comments in order to generate an edge probability matrix. Finally, in **Step 3**, we use the Chu-Liu-Edmonds algorithm (a Maximum Spanning Tree algorithm) to take this edge probability matrix and use it to generate the structure of the most probable ASG based on the stance detection model's predictions. We will describe each of these three steps in more detail below.

Step 1: An individual node summary is generated from each comment using a fine-tuned text summarisation model. The summarisation models are fine-tuned using the hyperparameters described Section 6.6.2. The fine-tuning dataset for this model is a reformatted version of Debatabase-ASG where the inputs and outputs are individual comments paired with individual summaries (with an identical train-val-test split).

Step 2: For every pair of comments, in a tree, u and v, one of the following is true: u attacks v, u supports v, or there is no directed relation from u to v (either because they are not linked, or because v in fact links to u)[14]. To represent these three possibilities, we use the labels "support", "attack", or "no relation".

To generate fine-tuning data for this problem, we again reformat Debatabase-ASG, so that the inputs to the models are in the format "Parent Comment: <Comment Text> Child Comment: <Comment Text>" and the outputs are a single label from the set {"support", "attack", "no relation"}. Our reformatted dataset for this subtask contains all support and attack relations from Debatabase-ASG as well as a random sample of the "no relation" pairs; the class distribution is "support": 2477, "attack": 1864, "no relation": 3000.

At inference time during the pipeline process, we use the LLM to approximate an edge probability matrix. This matrix represents the probability that node i is a child of node j, for all $i, j \in V, i \neq j$, where V is the set of nodes in the graph. We make the assumption that the probability of a link existing can be approximated by the largest softmax value generated by the LLM for each of the potential link relations, i.e. $\max\left(z_{\text{support}}(u,v),\ z_{\text{attack}}(u,v)\right)$ (where z is the softmax vector output by the LLM).

[14] It might be intuitive to think that if comment v attacks comment u, then u also attacks v. While that might be true in a literal semantic sense, in that u contradicts v, it is not true pragmatically — for many, if perhaps not all, comment-reply pairs. Many of the "counterpoint" comments contain explicit or implicit references to the comment that they are replying to, making it clear that v attacks u and not vice-versa.

Step 3: We then run the Chu-Liu Edmonds algorithm (Chu, 1965; Edmonds et al., 1967) over this node-pair matrix to generate a single tree. This algorithm is guaranteed to find the spanning tree with the highest weights, which can be interpreted as the set of potential edges with the highest overall model confidence.

Depth-1 Setting: In the depth-1 setting, we can take a more straightforward approach than the depth-2 setting, because we know the graph structure *a-priori*. Only two steps are required; firstly, we run the summarisation model over each comment, as in **Step 1** above. Next, we run the stance detection model as in **Step 2**. However, since all edges are known, we can simply run the model over each edge (which correspond to every comment linked to the root note), and apply the predicted "attack" or "support" label to the graph.

6.6.4 End-to-End Models

The fine-tuning process used for the end-to-end models was the same as that for the two types of model used in the pipeline, except for the data used: in the case of the end-to-end models, we used the entire comments section as an input, and for the output we use a text format that shows both the summaries and the relations between summary nodes (as illustrated in Figure 6.3).

Prior to evaluating the output of these models, we first post-process it as described in Section 6.5.

6.6.5 In-Context-Learning

In-Context-Learning (ICL) (Brown et al., 2020) is a general paradigm in NLP in which a model is not specifically fine-tuned on the target task, but instead is simply shown one or more examples of the task in question as part of the prompt that is passed to the model at inference time. ICL often relies on latent capacities of the model that have not been specifically trained for, but may exist due to the model having been trained on analogous tasks. It is therefore an interesting and open question to what extent these capacities may exist for our novel task.

We test two different In-Context-Learning strategies for the two larger models, Llama-2-7b and GPT-3.5. We abbreviate these two strategies **1P** (one-prompt), and **2P** (two-prompt). We will address these both in turn below.

Example 6.2: The format used in the one-prompt (1P) approach.

Comment thread: <COMMENT THREAD A>
Argument Graph: <ARGUMENT GRAPH A>

Comment thread: <COMMENT THREAD B>
Argument Graph: <ARGUMENT GRAPH B>

Comment thread: <COMMENT THREAD C>
Argument Graph: <ARGUMENT GRAPH C>

Convert the comment thread below into an argument graph, using the formatting shown in the three examples above

Comment thread: <TARGET COMMENT THREAD>
Argument graph:

One-Prompt Strategy In the one-prompt (**1P**) strategy, we use one prompt to convert the comment thread directly into an argument graph. Example 6.2 shows how the prompts were formatted for this task. Each prompt contained three examples of inputs and outputs (represented in the example as <COMMENT THREAD A>, <ARGUMENT GRAPH A>, etc.), followed by an instruction, and finally the target comment thread to be processed. The inputs and outputs were formatted as in Figure 6.3.

Example 6.3: The format used in the two-prompt (2P) approach

Prompt 1:
Comment thread: <COMMENT THREAD A>
Comment Summaries: <COMMENT SUMMARIES A>

Comment thread: <COMMENT THREAD B>
Comment Summaries: <COMMENT SUMMARIES B>

Comment thread: <COMMENT THREAD C>
Comment Summaries: <COMMENT SUMMARIES C>

Create a summary of each comment in the thread below, using the formatting shown in the three examples above

Comment thread: <TARGET COMMENT THREAD>
Comment summaries:

. . . . . .

Prompt 2:
Comment summaries: <COMMENT SUMMARIES A>
Argument graph: <ARGUMENT GRAPH A>

Comment summaries: <COMMENT SUMMARIES B>
Argument graph: <ARGUMENT GRAPH B>

Comment summaries: <COMMENT SUMMARIES C>
Argument graph: <ARGUMENT GRAPH C>

Convert the list of summaries below into an argument graph, using the formatting shown in the three examples above

Comment summaries: <TARGET COMMENT SUMMARIES>
Argument graph:

Two-Prompt Strategy The two-prompt (**2P**) strategy separates the task into two subtasks: firstly, summarisation, and secondly, detecting relations and relation types. To carry this out, we use the two prompts whose formatting is shown in Example 6.3. The first prompt transforms the input to an intermediate representation formatted as in Figure 6.4, in which the comments have been summarised but the relations between them are absent. The second prompt transforms this representation into the same textual representation of the argument graph as in the 1P approach.

6.7 Results

We used three different metrics in this study, which we describe in more detail in Chapter 5, Section 5.4. These include the ROUGE metric, which is intended to evaluate the correctness of the textual summaries generated, the Stance-F1 metric, which is intended to evaluate the correctness of the support/attack relations, and the Graph Edit Distance (GED) metric, which evaluates the correctness of the generated graph structure.

Our full results are displayed in Table 6.5. Note that in the flat tree setting, the pipeline models are guaranteed to get a "perfect" GED score of zero because we assume that the flat structure is known *a-priori*. In contrast, the TANL models, which do not have access to this information, sometimes generate an incorrect number of summary nodes.

Model	Depth-1 tree			Depth-2 tree		
	ROUGE-2 ↑	Stance-F1 ↑	GED ↓	ROUGE-2 ↑	Stance-F1 ↑	GED ↓
Pipeline Models						
Baseline	0.2263	0.3291	**0.0**	0.3787	0.3247	3.8
Longformer	0.1981	0.1952	0.0	0.1586	0.2461	12.476
Phi-2	0.3042	0.7238	0.0	0.4629	0.2224	12.597
LLaMA-2	0.3106	0.6931	0.0	0.4541	0.2505	8.25
End-to-end (ICL)						
LLaMA-2 (1P)	0.1889	0.3377	7.2667	0.0863	0.1528	10.2167
LLaMA-2 (2P)	0.1729	0.3491	6.6174	0.0794	0.1481	10.5307
GPT-3.5 (1P)	0.3239	0.856	1.8333	0.2577	0.7909	3.0167
GPT-3.5 (2P)	0.2942	0.7623	1.6539	0.2303	0.7060	2.7118
End-to-end (fine-tuned)						
Longformer	0.1864	0.2217	9.6949	0.1833	0.1943	8.4667
Phi-2	0.3597	0.9093	0.9333	0.546	0.7825	2.5667
LLaMA-2	**0.3953**	**0.9481**	0.2	**0.5829**	**0.87**	**1.45**
GPT-3.5	0.3137	0.9357	0.5667	0.5422	0.8333	1.85

Table 6.5: Results of the LLMs tested. Note that up-arrow indicates that higher scores are better, down-arrow indicates that lower scores are better.

Pipeline vs. End-to-End (RQ 6.1): The end-to-end approach substantially outperforms the pipeline approach, across all models used and for both depth-1 and depth-2 trees, with the exception of the GED for flat trees. The explanation for TANL performing better is two-fold: first, error propagation is a known limitation inherent in a pipeline approach. Moreover, the TANL model has access to global information about the debate when making a decision, rather than only having the local information about pairs of comments.

Tree Depth (RQ 6.2): Figure 6.5 compares performance on the Graph Edit Distance metric across the two different tree depths. It is clear that all models struggled with the depth-2 trees more than the depth-1 trees, with the exception of the end-to-end Longformer, which performed poorly on both.

The pipeline models' score of zero across all depth-1 trees is due to the fact that the graph structure is known *a-priori* (all nodes link to the root), and the algorithm for generating the node structure is deterministic. In contrast, the end-to-end models, even the better performing Phi-2 and Llama-2, occasionally produced errors involving omitting nodes from the input or adding ("hallucinating") additional ones. However, the end-to-end models showed a much better performance than the pipeline models did on depth-2 trees, with the fine-tuned version of Llama-2-End to End, for instance, having a GED of 1.45, compared to 8.25 for Llama-2-pipeline.

One possible explanation for this performance improvement is that the end-to-end models have access to the entire comment thread when making a decision about where to place a node, compared to the pipeline model which compares comment pairs individually. The model may erroneously assign a high confidence to a pair of nodes u and v having a connection, for instance, if it is unaware that there is another node q that is more obviously related to u.

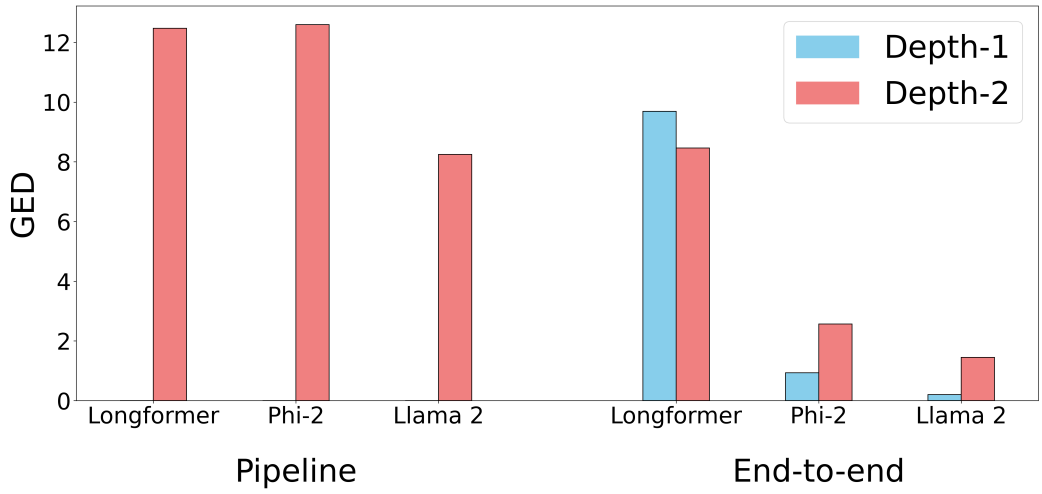


Figure 6.5: A comparison of performance on the Graph Edit Distance metric for the fine-tuned models (both fine-tuned and end-to-end) across the two different tree depths

Fine-tuning vs. Prompting (RQ 6.3): Furthermore, as expected, fine-tuned models outperformed the ICL models. The two variations of prompt-only Llama-2 models which we tested failed to beat the baseline. While the prompt-only GPT-3.5 model did beat the baseline, it still performed worse than all of the fine-tuned models across all metrics, with the exception of the Longformer. The comparatively poor performance of the ICL models can perhaps be explained by the fact that this task, and even the sub-tasks in the two-stage prompting solution, differ greatly from anything that the models have likely seen in their training data.

The results comparing the one-prompt (1P) vs the two-prompt (2P) strategy were inconclusive. For GPT-3.5, which was the better-performing model on the prompting task, it appears that the 1P strategy performed better than 2P strategy on ROUGE and Stance-F1, while the 2P strategy achieved better results with the GED metric. However, investigation using a wider range of models would be required to confirm this.

Base Model Selection (RQ 6.4): In general, the performance of the fine-tuned models seem to positively correlate with their size, as we would expect. While the Longformer model failed to beat the baseline, the other three larger models tested scored substantially higher than the baseline across all metrics. One partially surprising result is that the fine-tuned TANL GPT-3.5 slightly underperformed compared to Llama-2, but this may simply be due to the fact that we did not perform a hyper-parameter search on this model, owing to resource constraints.

6.7.1 Qualitative Analysis of Model Outputs

Tables 6.6 and 6.7 show two separate types of errors produced by the best performing model (Llama-2 fine-tuned end-to-end). Table 6.6 shows an example in which the model has made correct predictions with respect to the stances of nodes in the generated ASG, but has generated an incorrect graph structure. Table 6.7 shows the opposite error type: the graph structure has been predicted correctly, but there are several errors in the predicted stance due to incorrect support/ attack labels.

6.8 Conclusions

We have proposed a novel task in Argument Mining, ASG Parsing, which involves creating a graphical summary of an argumentative dialogue and have created and made publicly available a dataset that supports this task. We compared two main approaches to this task: a three-stage pipeline approach using separate summarisation and stance detection models, and an end-to-end approach. The end-to-end approach outperformed the pipeline solution across the board, showcasing LLMs' ability to address complex tasks that are

intuitively framed as separate tasks. Additionally, we looked at increasing the depth of the summary graphs to a depth-2 tree, and showed that this produces a dip in performance across all models. This result shows that despite the great progress made by LLMs in recent years, they may still struggle to deal with producing hierarchical structures.

6.9 Limitations and Future Work

Prompting strategies We have only attempted two different prompting strategies on this task, and therefore it is a strong possibility that more suitable prompts exist for this task. Prompt tuning, (Li and Liang, 2021) a technique in which a task specific "prefix vector" is learnt instead of updating the model weights, could be tested in future work.

Dataset Limitations The dataset which we produce in this chapter is a first attempt at generating an ASGP dataset. To investigate whether this task was feasible using LLMs at their current stage of development, we produced a resource which was somewhat simplified in several aspects, when compared to realistic argumentative debate, due in part to the limitations of the *Debatabase* data. These simplified aspects include:

- **Formal style:** Since the text is curated and edited by professional debaters, the style is much more formal and well-organised than can be expected in most spontaneous arguments, whether they occur online or face-to-face. This is likely to make the text more easily comprehensible, and therefore easier to interpret for a language model.
- **Shallow trees:** Due to the way that the data on idebate.net is formatted, the maximum depth of tree that we can extract is two, i.e. two levels

down from the root. However, as we have shown in our analysis of comments from the SENSEI dataset in Chapter 1, Section 1.3.2, the depth of argument trees is frequently deeper than this due to the occurrence of nested arguments (arguments may attack the premises of arguments which attack the premises of other arguments, and so on).

- **Known main topic:** As we have formulated this task, the main topic of each thread is known and is part of the input to the model. However, in a more realistic scenario, debate may break out between two or more people without any of the interlocutors explicitly stating what the "debate topic" is. Therefore, it may be more realistic to introduce a task in which one must first predict the debate topic and then generate an ASG with that topic as a root; this will inevitably be a more challenging task.

- **Binary debates**: Due to the format of competitive debate, which is an argument between two competitors, all comments are polarised, either being "for" or "against" the main topic. This is often not the case in real world dialogues; for example, commenters can be ambivalent on a given proposition, or simultaneously attack one proposition in an interlocutor's statement while defending another.

For these reasons, the dataset examined in this task was somewhat artificial. Future work should attempt to apply the models examined here to argumentative comment "in the wild". We will turn to this task in Chapters 7 and 8.

Main topic: This House believes western universities should only place satellites in states that respect free speech.	
Gold standard ASG	Llama-2 Generated ASG
Comment 1 (supports main topic): The academic tradition of the West	Comment 1 (supports main topic): Universities are beacons of free speech
Comment 2 (supports main topic): Maintaining the value of the degree	Comment 2 (supports main topic): Universities are service providers
Comment 3 (attacks main topic): 'Separation of town and gown'	Comment 3 (attacks Comment 2): There are two parties involved in this interaction, the state and the university.
Comment 4 (attacks Comment 2): Employers measure degrees by the academic results they indicate.	Comment 4 (attacks main topic): Employers measure degrees by the academic results they indicate

Table 6.6: Example output from the Llama-2 End-to-end fine-tuned model, illustrating errors in the predicted tree structure.

Main topic: This House would replace Trident	
Gold standard ASG	Llama-2 Generated ASG
Comment 1 (supports main topic): Can we rely on US nuclear umbrella?	Comment 1 (supports main topic): The UK needs its own nuclear deterrent
Comment 2 (attacks main topic): Trident is not an independent weapons system	Comment 2 (supports main topic): The UK does not have an independent nuclear deterrent
Comment 3 (supports main topic): Deterrence is still necessary.	Comment 3 (attacks main topic): Trident is still necessary
Comment 4 (supports main topic): Trident allows the UK to maintain its global status	Comment 4 (supports main topic): Losing trident would be a loss of status
Comment 5 (attacks main topic): The UK should encourage others to reduce their own stocks of nuclear weapons.	Comment 5 (supports main topic): The UK is obliged to disarm under the NPT

Table 6.7: Example output from the Llama-2 End-to-end fine-tuned model, illustrating errors in the predicted node stances.

Chapter 7

End-to-end Techniques for Argument Structure Parsing

7.1 Introduction

In Chapter 6 we explored the application of several LLM-based techniques to our proposed task of Argument Summary Graph Parsing (ASGP). We demonstrated that end-to-end models work significantly better than pipeline models for ASGP.

The task of Argument Structure Parsing (ASP) is well-established task in the field of Argument Mining, and shares a number of similarities with ASP (we have reviewed the task in detail in Chapter 3). However, very little work has been done on end-to-end models for ASP. Before we move on to pursuing ASGP on more realistic datasets (the topic to be addressed in Chapters 8 and 9) we will pause to explore the effectiveness of the end-to-end approach that we have developed for the ASGP task in ASP.

While numerous approaches have been proposed for ASP, there are significant advantages to treating the entire problem as a text-to-text task learned end-to-end. This approach eliminates the need for task-specific engineering of model architectures or loss functions. Using this general task framing, the

Argument Graph which is the output of an ASP model can be treated like any other textual representation, removing the necessity for custom, task-specific loss functions or model heads.

To our knowledge, only one prior study (Kawarada et al., 2024) has investigated Argument Structure Parsing as a text-to-text task. This highlights a clear opportunity for further research in this area. Specifically, our aims in this chapter are to compare alternative methods of encoding the output graph in text format, and different LLMs. The models compared are Flan-T5, (Chung et al., 2022) an encoder-decoder model used by Kawarada et al. (2024) and LLaMA-3, (Grattafiori et al., 2024) a more recent decoder-only model.

7.2 Background

In this section we will firstly describe the task addressed in this Chapter, Argument Structure Parsing (Section 7.2.1), before describing the dataset used, the Argument Annotated Essays Corpus (Section 7.4.1), and finally briefly reviewing previous approaches to the task (7.2.2).

7.2.1 The Argument Structure Parsing Task

The task of Argument Structure Parsing consists of taking an argumentative text (typically an essay), and using it to generate a graph representation showing the Argumentative Discourse Units (ADUs) present in the text, as well as the relationships between them. ADUs are text spans representing components of an argument, which we have defined in more detail in Chapter 2, Section 2.3.

Stab and Gurevych (2014a) divide the task of Argument Structure Parsing into four sub-tasks, which we have described in detail in Chapter 3, Section 3.1. For clarity, we briefly reiterate them here:

1. Argument Component Identification (ACI) is the task of splitting the text into "Argumentative Discourse Units" or ADUs, which correspond, for example, to a Claim or a Premise.

2. Argument Component Type Classification (ACTC) is the task of classifying ADUs into either a Claim, a Premise or a Major Claim. In Stab and Gurevych's scheme, every essay has a single "Major Claim" and every paragraph in the essay has a single "Claim" which is supported or attacked by one or more "premises".

3. Argument Relation Identification (ARI) is the task of identifying (directed) links between Premises, Claims, and Major Claims. There are restrictions on these link types, shown in Table 7.2.

4. Argument Relation Type Classification (ARTC) classifies the links identified by the ARI component as either "support" or "attack".

Each of the latter three subtasks corresponds to an evaluation metric, which we have explained in Chapter 3, Section 3.2.

7.2.2 Previous Approaches to Argument Summary Parsing

In Chapter 3, Section 3.4, we have reviewed the previous approaches taken to ASP on the AAEC. As explained in more detail in that section, there are two main model types, "ADU-level", which take pre-segmented ADUs as input, and "end-to-end", which take as input raw text or a list of tokens. Table 7.1

provides a list of the previous works which we use for comparison systems across the two categories.

	Model Name	**Model Type**
ADU-level Models	Joint ILP (Stab and Gurevych, 2017)	Support Vector Machine with hand-crafted features
	Span LSTM (Kuribayashi et al., 2019)	ADU-level MST parser using LSTMs
	Bert-Trans (Bao et al., 2021)	Neural transition-based ADU parser using BERT
	CU-MAM (Gemechu and Reed, 2025)	Graph Neural Network with Multi-Task Learning
End-to-end Models	LSTM-ER (Eger et al., 2017)	Token-level LSTM dependency parser
	ST Model (Morio et al., 2022)	Longformer-based joint span-segmentation and ADU-level MST parser
	Flan-T5-XXL (Kawarada et al., 2024)	End-to-end text-to-text model based on Flan-T5

Table 7.1: Previous models which have been used for ASP on the AAEC

7.2.3 Previous approaches to post-processing and evaluation

In this Chapter, we use the use the standard metrics that researchers have previously applied to the AAEC corpus (see Section 3.2.3). However, we differ in the post-processing techniques for the model output that we chose to apply before calculating metrics. Here, we will explain how this has been tackled previously, and explain our rationale for using an alternative approach.

As we explained in Section 3.2.3, two different sets of metrics are used on this task, one used to evaluate ADU-level models, originating in Stab and Gurevych (2014b) and the other, used for end-to-end models, originating in Eger et al. (2017).

Running evaluation metrics on ADU-level models is straightforward due to the fact that the units being classified are known beforehand and therefore the model makes simple classification decisions which are easy to compute a score over. However, for the end-to-end models, a complication is added: the ADU spans predicted by models do not necessarily match the spans in the gold standard.

Eger et al. (2017), following the previous work of Persing and Ng (2016), find a solution to this matching problem which works only for models which work by classifying tokens. In their work, an ADU in the model output counts as "matching" an ADU in the gold standard if ≥ 50% of tokens in a predicted and gold ADU overlap. They release with their paper a script implementing this technique.

The token overlap approach, however, is only adequate if the gold and predicted output are still aligned at a token level. As we have seen, models such as Kawarada et al. (2024) can output free text, and therefore the assumption that two ADUs containing the same set of tokens are similar is unwarranted. Kawarada et al. (2024) opt to solve this by using the Needleman-Wunsch algorithm (Needleman and Wunsch, 1970), a sequence aligner, to first globally align the output from their model with the input, before using the script from Eger et al. (2017) to perform evaluation.

In this chapter, we instead formulate ADU alignment as a bipartite set-matching problem subject to a hard constraint on normalised Levenshtein distance between ADU spans. This formulation is preferable to Needleman-Wunsch

in our setting given its interpretability, and the fact that the results it gives are invariant to permutations in the order of ADUs in the reference text. We describe our approach in greater detail in Section 7.5.

7.3 Research Questions

In this chapter, we address three research questions, enumerated below:

RQ 7.1 — Can an end-to-end model outperform an ADU-level model if segment information is included? We compare the performance of our end-to-end approach with previous pipeline approaches to the same task. Previous work in ASP has been divided into two separate model types: end-to-end models and "ADU-level" models, which rely on a prior segmentation step, in which the text is split into Argumentative Discourse Units (ADUs). Authors typically report the results of the models using *oracle* segmentation, i.e. using the gold-standard ADU splits which are given in the corpus. Therefore, to try and give a fair comparison between the models that we test here and the state-of-the-art ADU-level models, we experiment with an "oracle" setting in which we give the end-to-end model access to the gold-standard splits.

RQ 7.2 — How do the two output formatting variations we investigate affect model performance? Secondly, we investigate the formatting of the model output. Kawarada et al. (2024) represented the output by returning a copy of the essay in the input annotated with a markup language-like syntax. Alternatively, the output may be formatted in a similar way to the format we chose for Argument Summary Graph Parsing in Chapter 6, i.e. as a numbered list of all ADUs contained within the essay, with the relation to its parent shown in parentheses.

RQ 7.3 — Which perform better at ASP: encoder-decoder or decoder-only models? Finally, we compare the performance of two different LLMs on this task: Flan-T5-XXL (previously used by (Kawarada et al., 2024) and LLaMA-3, a more recently released LLM of comparable parameter size. Our motivation is to investigate whether the decoder-only architecture of LLaMA-3 — the most prevalent type of LLM architecture among state-of-the-art models in recent years (Qorib et al., 2024) — can perform as well as the encoder-decoder architecture of Flan-T5.

7.4 Dataset and Formatting

In this section, we first provide background on the corpus which we use in this chapter (the Argument Annotated Essays Corpus or AAEC) followed by a description of the formatting conventions which we will use in the experiments described in Section 7.6.

7.4.1 The Argument Annotated Essays Corpus

For this study, we chose to use the Argument Annotated Essays Corpus (Stab and Gurevych, 2014a). It is one of the most well-studied corpora in Argument Structure Parsing, and is therefore a useful reference corpus to compare how well our chosen method performs against a variety of previously used techniques.

The Argument Annotated Essays Corpus, or AAEC, is a collection of 402 Essays of the type that second-language learners of English are often assigned to write in secondary school. Each essay argues for or against a certain controversial proposition, for example "The option to work or study from home is an advantage", or "Modern communication technologies benefit all people".

Each essay has been annotated with three different components: firstly, it has been segmented into ADUs (Argumentative Discourse Units). Each ADU span has one of three labels: MC (MajorClaim), C (Claim), P (Premise). There are additional non-ADU spans which are labelled O (Outside/ non-argumentative). Each essay contains an average of 15 ADUs.

From	Relation	To
Claim	For/Against	MajorClaim
Premise	Support/Attack	Claim
Premise	Support/Attack	Premise

Table 7.2: The set of permitted relations in the AAEC

As well as ADUs, the AAEC contains annotations for relations between the components. Four different types of relations are annotated: For, Against, Support, and Attack. There are restrictions on which components may be linked to which other components, as well as the possible relations between them. These are illustrated in Table 7.2. As the Table indicates, the main difference between the For/Against relations and the Support/Attack relations is that one pair of relations is used to annotate *Claim* → *MajorClaim* links, while the other is used for *Premise* → *Claim* and *Premise* → *Premise* links. Therefore, following the majority of researchers working on the AAEC (e.g. Stab and Gurevych 2014a; Kawarada et al. 2024), we have decided to treat these two pairs of relations as identical for scoring purposes, mapping *For* to *Support* and *Against* to *Attack*.

7.4.2 Formatting Used in Our Experiments

Our end-to-end framing of the task requires that we pre-process the corpus into a form that is amenable to this method. To address the first and second Research Questions of this chapter, we use two different variants each of input

and output formatting, which we describe in turn below.

Input Formatting: The two different variations of input formatting are the *oracle-ADUs* setting and the *no-oracle-ADUs* setting. In the *no-oracle-ADUs* setting, the essay is simply passed to the model as a single string, unchanged from the original corpus. In the *oracle-ADUs* setting, special symbols are added to the input to indicate the start and end of all text spans which were annotated as ADUs. The start of ADUs is indicated by an exclamation mark between parentheses, and the end is indicated by a pipe symbol, as shown in Table 7.1.

Example 7.1: Input formatting in the oracle-ASG setting.

Another reason is that (!)most of young people prefer to have meals at restaurants |because (!)they want to invest more time in their work or study to improve themselves |

Output Formatting For the output formatting, we compare two different ways of encoding the graph in text form. The first of these is what we call "markup style" — this is the formatting which Kawarada et al. (2024) use in their work. Table 7.3 shows the formatting patterns used for each ADU type. In the table, `{stance}` is a member of the set {`for, against`}, and indicates the stance of the Claim towards the MajorClaim. `{relation}` is a member of the set {`attacks, supports`}. The output corresponding to an essay fragment in the markup style is shown in Example 7.2.

The second formatting style tested is what we call "list style", our alternative proposal. In this style, all ADUs appear in a numbered list, similar to the formatting that we used for AGSP in Chapter 6. Table 7.4 describes the formatting used. In the table, `{adu_code}` and `{parent_code}` refer to the

ADU Type	Formatting
MajorClaim	`[ {adu_text} | MajorClaim ]`
Claim	`[ {adu_text} | Claim {stance} ]`
Premise	`[ {adu_text} | Premise {relation} = {parent_text} ]`

Table 7.3: Formatting used for the outputs in ”Markup style”.

Example 7.2: Output formatting in “Markup” style (example from Kawarada et al. 2024).

As far as I am concerned, [we should do our parts to do what ever it takes to protect old buildings, letting next generations still have a chance to look at them | MajorClaim]. Here are some reasons and examples to illustrate my viewpoint. [Historic buildings have intrinsic values | Claim for], because [they are significant symbols of a city | Premise | supports = Historic buildings have intrinsic values]. [Old buildings are reminders of a city ’ s culture and complexity | Premise | supports = Historic buildings have intrinsic values].

ADU type followed by an integer, such as ”Premise 1” or ”Claim 2”. The output corresponding to an essay fragment in this style is shown in Example 7.3. Comparing it to the markup style described above, we can see that the list style representation is more compact, as it lacks repeated text spans. In the markup style, the entire text span corresponding to an ADU’s parent is repeated within the square brackets surrounding that ADU. In contrast, the list style refers to each ADU with a unique code such as “Claim 1”, making the representation more compact.

ADU Type	Formatting
MajorClaim	`MajorClaim: {adu_text}`
Claim/Premise	`{adu_code} ({relation} {parent_code}): {adu_text}`

Table 7.4: Formatting used for the outputs in ”List style”.

Example 7.3: Output formatting in the "List style".

MajorClaim: we should do our parts to do whatever it takes to protect old buildings, letting next generations still have a chance to look at them.
Claim 1 (supports MajorClaim 1): Historic buildings have intrinsic values.
Premise 1 (supports Claim 1): they are significant symbols of a city.
Premise 2 (supports Claim 1): Old buildings are reminders of a city's culture and complexity.

7.5 Evaluation

Evaluating model output is a three-stage process. The stages are as follows.

1. Using regular expressions, parse the model output to obtain a graph, G_{pred}.
2. Relabel the nodes in the G_{pred} to align with the gold standard, according to ADU similarity
3. Evaluate the relabelled graph using the metrics listed in Section 7.5.3.

In the subsections below, we will describe each of these three steps in order.

7.5.1 Output Parsing

Regular expressions are used to scan the raw textual output of the model and extract the predicted argumentative discourse units (ADUs) and their relations. As the output is processed sequentially, we iteratively construct the predicted graph $G_{\text{pred}} = (V_{\text{pred}}, E_{\text{pred}})$, where each extracted ADU gives rise to a node in V_{pred}, and each explicitly stated relation between ADUs gives

rise to a directed, labelled edge in E_{pred}. The resulting graph is a labelled directed graph and need not be connected, as the model may produce multiple independent argument structures within a single output.

7.5.2 Node Relabelling

The metrics used in this chapter, which we have defined in Section 3.2 are calculated using ADU labels, under the assumption that the labels in G_{gold} and G_{pred} refer to the same underlying entities. In practice, however, the graph G_{pred} produced by our models may use labels that do not correspond directly to those in the gold standard. It is therefore necessary to *relabel* G_{pred} with respect to the gold-standard ADUs before computing these metrics.

The metrics assume a shared set of uniquely labelled ADUs across both graphs. For this assumption to hold, the following conditions must be met:

1. An ADU in V_{pred} and an ADU in V_{gold} share the same label if and only if they are identical or sufficiently similar.
2. Each ADU label in V_{gold} is assigned to at most one ADU in V_{pred}.
3. The alignment between V_{pred} and V_{gold} maximises overall similarity subject to the above constraints.

We address this by treating ADU relabelling as a matching problem between the gold and predicted node sets, V_{gold} and V_{pred}, which we solve using the Hungarian algorithm (Kuhn, 1955). Each node may be matched to at most one node in the other set, and nodes may also remain unmatched if no suitable correspondence exists.

Similarity between ADUs is measured using the normalised Levenshtein distance (Levenshtein et al., 1966) computed over word-tokenised text. We

construct a cost matrix C in which each entry C_{ij} represents the Levenshtein distance between node $i \in V_{\text{gold}}$ and node $j \in V_{\text{pred}}$.

We apply a threshold of 0.2, such that gold–pred ADU pairs with a distance ≥ 0.2 are excluded from consideration by assigning them a large constant cost of 10,000. Since the Hungarian algorithm requires a square cost matrix, we pad C with dummy rows or columns as necessary to obtain a matrix of size

$$\max(|V_{\text{gold}}|, |V_{\text{pred}}|) \times \max(|V_{\text{gold}}|, |V_{\text{pred}}|).$$

Matching a node to a dummy row or column corresponds to leaving that node unmatched. The cost of such dummy matches is set to 100, which is higher than any valid Levenshtein distance but lower than the cost assigned to excluded pairs. This encourages valid gold–pred matches where possible, while allowing nodes to remain unmatched when no suitable match exists.

The resulting assignment[15] defines a partial mapping from V_{pred} to labels in V_{gold}. Predicted nodes that are matched inherit the labels of their corresponding gold nodes, while unmatched predicted nodes are assigned fresh, unique labels not present in V_{gold}.

7.5.3 Metrics

We use the metrics described in Section 3.2 to evaluate model performance. For comparability with past results, we use the Component-F1 (C-F1), Relation-F1 (R-F1) and Relation Type-F1 (T-F1) to evaluate models where oracle ADU segment information is provided, and Component Micro-F1 (C-Micro-F1) and Relation Type-Micro-F1 (T-Micro-F1) and in the no-oracle-ADUs setting.

[15] We use the implementation provided by `scipy.optimize.linear_sum_assignment`.

7.6 Experiments

Table 7.5 illustrates the three different experimental variables that we address, and their different conditions, corresponding to the three Research Questions posed in Section 7.3.

We have already covered the input and output formatting variations in Section 7.4.2. In the remainder of this section, we turn to describing the two model types used and the training procedure.

Variable	**Conditions**
Input format (**RQ 7.1)**	Oracle ADUs included, Not included
Output format (**RQ 7.2)**	Markup-style, Plain text
Model type (**RQ 7.3)**	Flan-T5, LLaMA-3

Table 7.5: Experimental variables and the conditions tested.

7.6.1 Model Training

We fine-tune two different models on this task, Flan-T5-XXL (Chung et al., 2024), and LLaMA-3 (Dubey et al., 2024). We chose to use Flan-T5-XXL to attempt to replicate the results obtained by Kawarada et al. (2024). LLaMA-3 was chosen due to the fact that T5 models have an encoder-decoder architecture, and we were interested to see if a decoder-only model could obtain comparable performance.

We carried out fine-tuning using eight NVIDIA V100 GPUs, each with 32GB of VRAM. Both of the models tested were too large to fine-tune in their entirety using this setup without using a parameter-efficient fine tuning (PEFT) technique. We chose to use QLORA (Dettmers et al., 2023), a technique which we

Feature	Flan-T5-XXL	LLaMA-3-7B
Number of Parameters	11 billion	7 billion
Architecture	Encoder-Decoder	Decoder-Only

Table 7.6: Comparison of Flan-T5-XXL and LLaMA-3-7B

have explained in Chapter 2 Section 2.6.

Hyperparameters were set by carrying out a separate grid search for each of the 8 combinations of experimental conditions ($2 \times 2 \times 2$) shown in Table 7.5. For both models, we fine-tuned for 8 epochs and employed early stopping with a patience parameter set to 2 epochs, meaning training was halted if the validation performance did not improve for two consecutive epochs. The grid search was conducted over three hyperparameters: LORA dropout (from the set $1e{-}1, 5e{-}1$), learning rate (from $1e{-}3, 1e{-}4$), and weight decay (from $1e{-}3, 1e{-}4, 1e{-}5$). The full list of parameters used in our models is shown in Appendix A, Section A.3.

7.7 Results

Tables 7.7 and 7.8 show the results obtained using the metrics listed in Section 7.5.3. The two tables compare to our results to those reported by other researchers, in the no-oracle span and oracle-span settings respectively. Our implementation of Flan-T5-XXL-Markup obtained different results from the original implementation in Kawarada et al. (2024); the C-F1 of our implementation was 2.5 points lower, whereas the R-F1 was 14 points higher.

ADU segment information (RQ 7.1) Table 7.8 shows the performance of the models in which oracle ADU spans were provided. The lower scores for ourend-to-end implementations compared to the SOTA comparison systems shows that even in the presence of perfectly detected ADU boundaries, end-to-end models still underperform at this task. This indicates that the higher performance of ADU-level models results from the architecture or model constraints chosen, and not just the fact that they have access to the added information of oracle span boundaries.

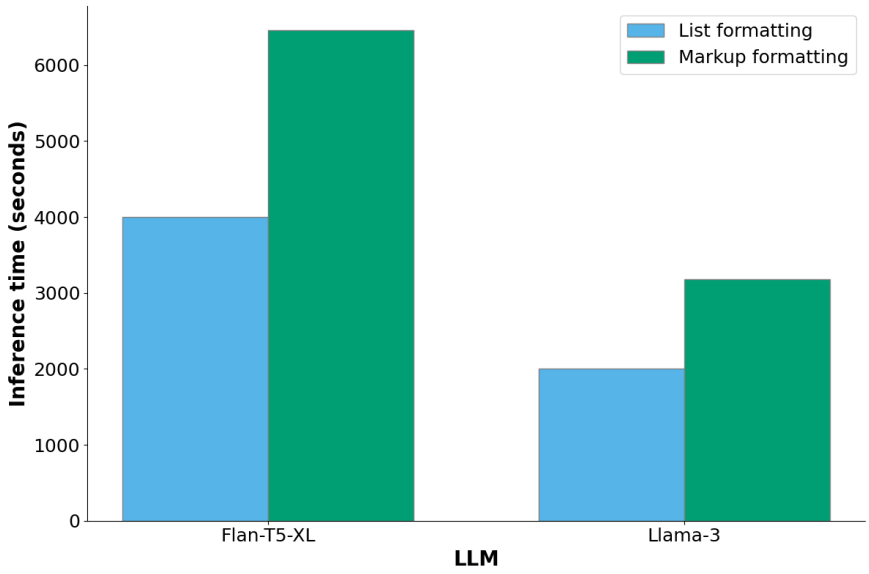


Figure 7.1: Comparison of inference time across the two models and formatting schemes that we used. Inference times refer to carrying out inference over the entire AAEC, using four H100 GPUs.

Output formatting (RQ 7.2) Across all metrics and models, markup formatting outperformed list formatting, albeit generally by a relatively small margin (the mean difference in F1 score across all model types and metrics is 7.3). A Wilcoxon signed-rank test showed that this difference is significant ($W = 0.00$, $p = .002$, $r = .89$).

Figure 7.1 shows a comparison of the inference time for the two different

models and formatting schemes that we have tested. Due to the more compressed textual output, inference time using list formatting was just under 60% of inference time with markup formatting. Considering this fact, it may sometimes be more practical to use list formatting as the output type for a text-to-text ASP model, even though list formatting models performed worse at the task.

Encoder-decoder vs. decoder-only (RQ 7.3) A two-sided Wilcoxon signed-rank test indicated that Flan-T5-XXL achieved significantly higher performance than LLaMA-3 (W = 5.00, p = .020, r = .73), reflecting a large effect size. However, the average difference in F1 was quite small (3.8 F1). This result shows that while an encoder-decoder architecture may still be more suitable for this task, decoder-only architectures can achieve similar results.

Method	C-F1-Micro	T-F1-Micro
LSTM-ER (Eger et al., 2017)	66.21	29.56
ST Model (Morio et al., 2022)	76.55	54.66
Flan-T5-XXL (Kawarada et al., 2024)	**80.15**	**61.19**
Our Implementation		
Flan-T5-XXL-list	76.05	42.10
Flan-T5-XXL-Markup	78.05	54.80
LLaMA-3 list	75.20	36.00
LLaMA-3 Markup	77.50	44.50

Table 7.7: Results for models with no oracle ADU spans

Method	C-F1	R-F1	T-F1
Joint ILP (Stab and Gurevych, 2017)	82.6	75.1	68.0
Span LSTM (Kuribayashi et al., 2019)	85.7	80.7	79.0
Bert-Trans (Bao et al., 2021)	**88.4**	82.5	81.0
CU-MAM (Gemechu and Reed, 2025)	87.1	**85.4**	**82.7**
Our Implementation			
Flan-T5-XXL-list (ours)	74.50	79.60	46.80
Flan-T5-XXL-Markup	81.90	80.90	66.50
LLaMA-3 list (ours)	73.90	78.50	41.80
LLaMA-3 Markup	81.10	81.70	50.90

Table 7.8: Results for models with oracle ADU spans provided

Output Analysis The results in Table 7.8 show that each of the models we trained, especially the best-performing LLaMA-3 Markup model, performed close to the state of the art at Argumentative Relation Identification (as shown by the R-F1 scores). However, all models performed somewhat worse at component type classification, with the component F1 score for our best model being around 0.08 below the state of the art. Figure 7.2 is a confusion matrix showing the misclassified ADUs. This matrix illustrates that our model mostly classified Major Claims accurately, while the biggest source of misclassification was confusion between Premises and Claims. This is not surprising, since there is only one Major Claim per essay, and they generally appear as the first ADU in the text.

7.8 Conclusions

In conclusion, we have replicated the results of Kawarada et al. (2024) that showed that end-to-end techniques are viable for the task of Argument Struc-

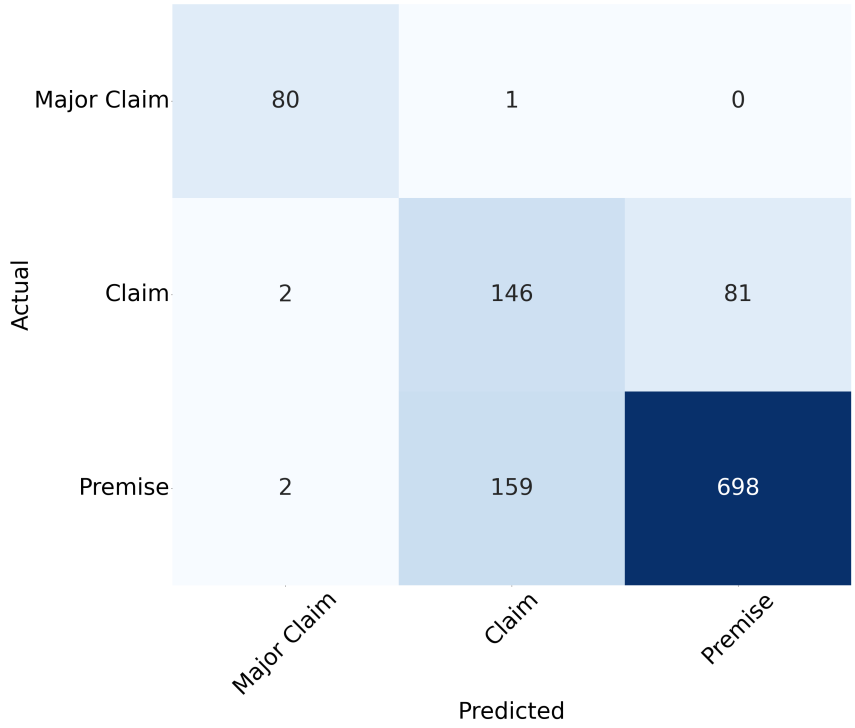


Figure 7.2: Confusion Matrix for ADU Classification for our best-performing model with Oracle Spans (LLaMA-3 Markup)

ture Parsing. Additionally, we have shown that both encoder-decoder (Flan-T5) and decoder-only (LLaMA-3) models are suitable for this task. We additionally trained a set of models for which oracle ADU span boundaries were provided, and found that these models underperformed when compared to the state of the art for oracle-ADU-span models. However, the gap between the current state-of-the-art and the results of our best performing model was rather small, suggesting that future developments can make text-to-text ASP a viable method.

7.9 Limitations and Future Work

In this chapter, we have addressed ASP on only a single dataset (AAEC), and using only two near-*state of the art* models. In future work, it would be interesting to see how well the text-to-text approach can generalise across datasets (e.g. on multi-party discussions or on texts with less formal structure than an essay).

The largest model that we have tested contained around 11 billion parameters. With less hardware limitations, it would be interesting to see how well a larger model could perform on this same task.

Decoder-only vs. encoder-decoder: In this chapter, we have shown that a single state-of-the-art decoder-only model performs comparably to a single state-of-the-art encoder-decoder model. In future work, it would be interesting to test a variety of models from the two classes to try and demonstrate more conclusively if one approach is superior to the other for this task.

Chapter 8

A Dataset for Argument Summary Graph Parsing of Online Comments

8.1 Introduction

In this chapter, we describe the process by which we created a new corpus for studying Argument Summary Graph Parsing on online news comment. The comments are sourced from the website of the British newspaper, *The Guardian*. The dataset that we create here was adapted from another publicly available dataset, SENSEI (Barker et al., 2016). We refer to our new dataset as SENSEI-ASG (SENSEI-Argument Summary Graph). [16]

In Chapter 5, we described our experiments in the task of Argument Summary Graph Parsing (ASGP), on a simplified and somewhat artificial debate dataset which we created using `debatabase.org`, a carefully curated website that is maintained by an international debating organisation.

Our motivation in creating this corpus is to create a more authentic, "naturalistic" resource for investigating debate summarisation when compared to the data source that we used previously. SENSEI consists of comments on newspaper articles, of both an argumentative and non-argumentative nature.

[16] Released publicly at `https://github.com/acidrobin/SENSEI-ASG`

To create our corpus, we filtered out an argumentative subset and labelled the support/attack relations between comments, using a procedure which will be described in this chapter.

This chapter is split into five main sections: we first describe the source corpus, SENSEI (Section 8.2), followed by outlining our aims in creating the new corpus (Section 8.3), and then describing our process for creating SENSEI-ASG (Section 8.4). Next, we discuss our obtained Inter-Annotator Agreement (Section 8.5). Finally we give some descriptive statistics and examples from the created corpus (Section 8.6).

8.2 Background

In contrast to the Debatabase-ASG dataset, our aim in creating this dataset was to develop a more "naturalistic" set of argumentative dialogues coupled with Argument Summary Graphs, which is derived from spontaneous arguments that people have engaged in online.

We chose to develop a dataset adapted from the SENSEI Annotated Corpus. This is a dataset derived from comments sections from the Guardian news website (`theguardian.com`). SENSEI was initially developed for NLP tasks involving the summarisation of online comment, but not for the specific task that we propose here (lacking, for instance, support/ attack annotations).

The original dataset, however, does have desirable criteria which made it amenable to developing a corpus for ASG generation:

1. Each of the comments has already been annotated with an associated summary which can be used in the ASG.

2. Each comment section has a number of manually annotated topics into

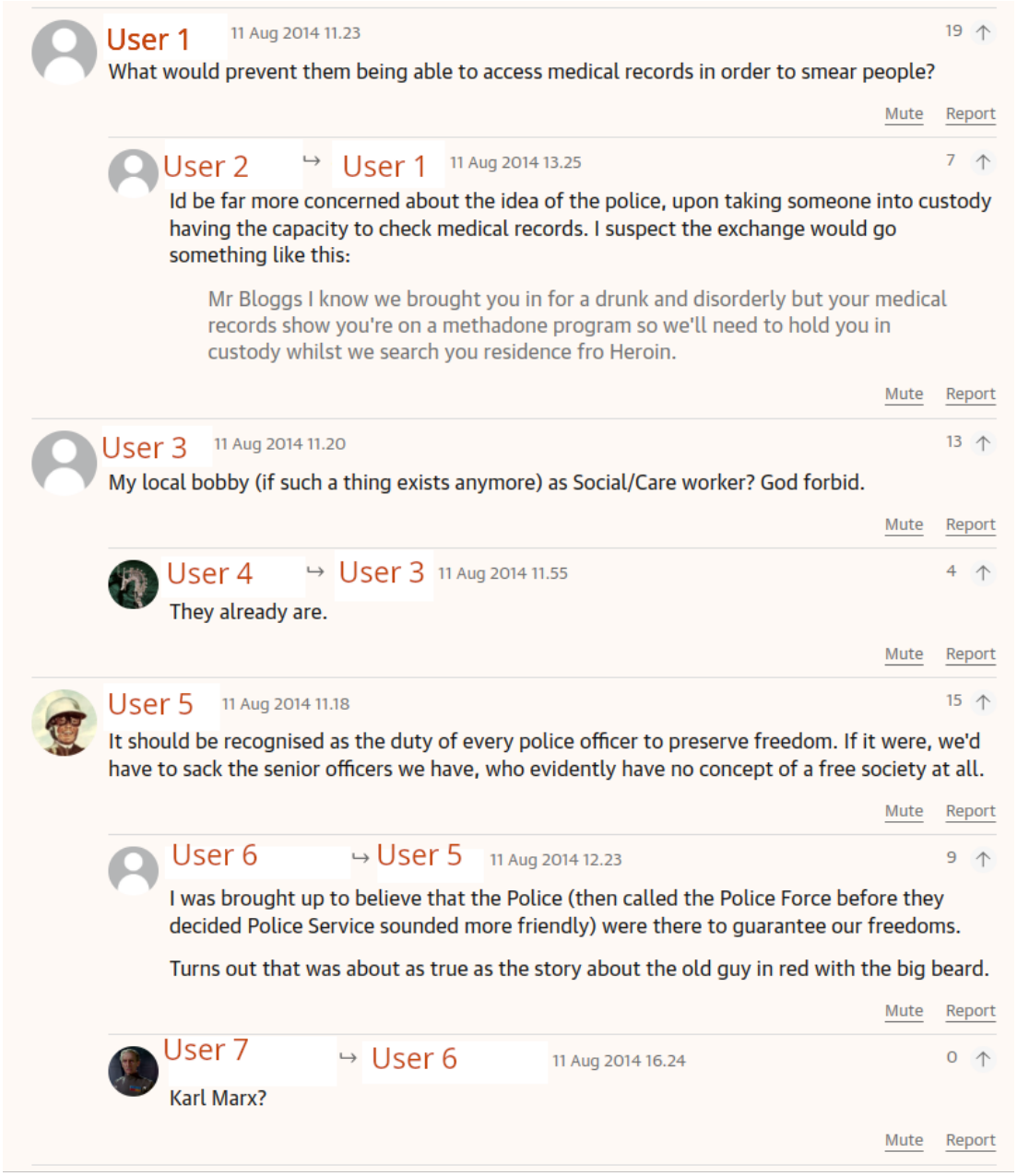


Figure 8.1: A sample of the comments which appear below news articles on the Guardian website. This example illustrates how the comments are tree structured, with responses appearing as indented blocks below the comment being replied to.

which the comments have been sorted. We can use these as a starting point to annotate "discussion topics" in a way that we describe in Section 8.4.

3. With respect to the content of the comments themselves, a reasonably large proportion are dedicated to argument/ discussion. This differs from other large-scale dialogue datasets (e.g. Chen et al. 2021) where argument appears to be more scarce.

8.2.1 Description of SENSEI

The SENSEI Annotated Corpus is a dataset containing both news articles and online comments from the Guardian news website (`theguardian.com`). In total, it contains 18 news articles coupled with the comments sections attached to these articles.

They have been annotated by a total of 15 reviewers, with every article being annotated by two reviewers each. Each comments section contains between 100 and 113 comments.

A number of different annotations are made available for each comment section; these include a textual summary of the entire comment section, summaries of comment subgroups, labels of topical clusters and summaries of individual comments. The most useful for our purposes are the latter two, as we use these to reduce the amount of work required in generating ASGs for each comment section. An example of these two annotation types is given in Figure 8.2.

GROUP: **Phone hacking**

Label for comment 29: *Police will easily be able to access information*

Label for comment 30: *Only 'normal' people would be vulnerable*

GROUP: **Vulnerable children**

Label for comment 31: *Social workers need a bigger involvement in child care so that police will not have to deal with incidents*

Label for comment 32: *Police having medical records will not help*

GROUP: **Medical conditions**

Label for comment 37: *Police should be more aware*

Label for comment 38: *People should have ID to say if they have a serious condition. That way police won't need access to records.*

Label for comment 39: *Hard to enforce people wearing a bracelet*

Label for comment 40: *Which conditions should be specified so police are aware?*

Figure 8.2: Example of the annotations available in the original SENSEI corpus. Text in bold indicates the names of GROUPS; all of the comments below the name are sorted into that GROUP. The texts in italics are summaries, or in the terminology of the SENSEI dataset, "labels", which summarise individual comments.

8.2.2 The SENSEI Annotation Process

The process by which the original corpus was annotated, as described in Barker et al. (2016) was as follows:

1. Annotators were assigned to different comments sections. Each comments section was annotated in its entirety by two independent annotators.

2. For each comment in the comments section, the annotator came up with a "label", or a short summary, describing the content of the comment. Annotators were given a high degree of freedom in how to do this, and the labels produced range from short phrases to multi-sentence summaries.

3. Annotators sorted each comment into "groups" that they created themselves. Once again, annotators were given a high degree of flexibility in how they carried out this step: there were few constraints either on the group names (these ranged from single words to longer phrases), or the size of the groups. In addition, they could also split the groups into multiple labelled subgroups. The only constraint was that each comment could be placed in one, and only one, group or sub-group.

There were two further steps carried out in the annotation process, which involve generating a more lengthy free-text summary of the entire comment section, but we omit those here since we do not use these parts of the corpus for our own purposes.

8.3 Aims of the Corpus

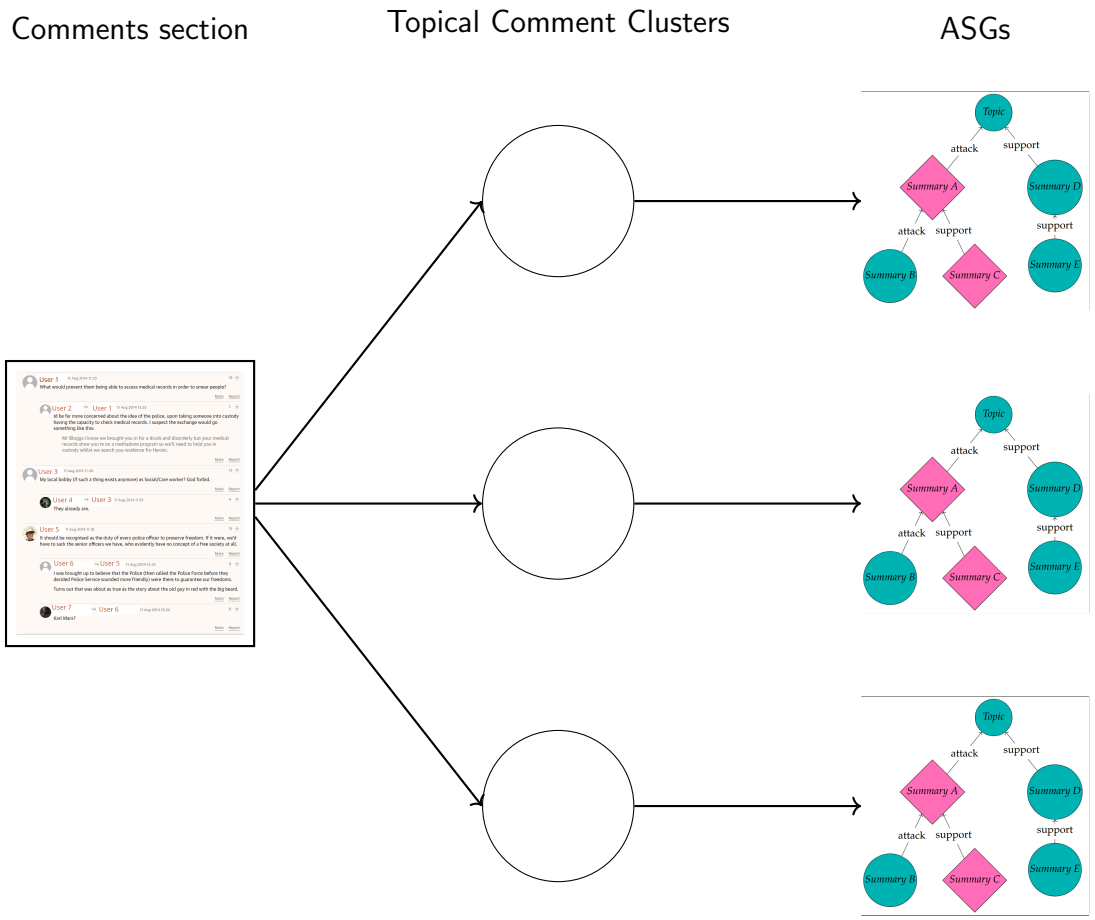


Figure 8.3: A simplified view of a proposed pipeline system to generate ASGs from a comments section

In Chapter 5, Section 5.3.4, we have explained a proposed pipeline approach for generating ASGs from a dialogical text. As we stated in that section, it is unrealistic for a single model to generate an ASG directly from a lengthy dialogue such as a comments section — among other reasons, because each of these comments sections may have multiple discussion topics.

We have therefore proposed that ASGP (the stage in which a graph is actually generated), be the last step in a pipeline which first involves, among other possible steps: (1) identifying comments which contain argument; (2)

clustering the comments by topic, and (3) identifying the main argumentative proposition which each cluster is discussing. A simplified view of the pipeline, showing only the comment clustering and ASGP stages, is shown in Figure 8.3.

Since our proposed task focuses on the final stage of this pipeline, we must create a gold standard for both the inputs to the task (the comment clusters) and the outputs (the ASGs). SENSEI already contains some topical comment clusters; however, as we will explain in Section 8.4 below, these clusters are not of the type that we would expect to be generated by an ideal comment clustering algorithm. In the next section, we explain how we use a partially automated process, as well as human annotation, to generate these gold standard comment clusters, along with the ASGs.

8.4 Our Corpus Creation Process

As explained in the previous section, the ASGP task is designed to be carried out on comment clusters around a single topic. Our annotation process therefore has two outputs:

1. Comment clusters, which are the **inputs** for the ASGP task
2. ASGs, the **outputs** for the ASGP task

This section describes how we used the SENSEI data to create our own corpus of argument-annotated data, SENSEI-ASG. This is fundamentally a manual annotation process, but also includes some automated steps. We used a pipeline process that takes advantage of several features of the existing annotations in SENSEI, including the clusters, and the per-comment summaries, as well as the reply structure that is present in the original Guardian comment sections. The overall process has eight steps, and is summarised in Figure 8.4.

As the diagram illustrates, the first two steps of the pipeline are used to generate "Input Comment Clusters", by which we mean the comment clusters that form the inputs to our new dataset. The remaining steps of the pipeline, Steps 3-8, describe the process by which we take the extracted comment clusters, and then perform several annotation steps to produce ASGs, the output of our dataset. We now turn to describing each individual step in the pipeline.

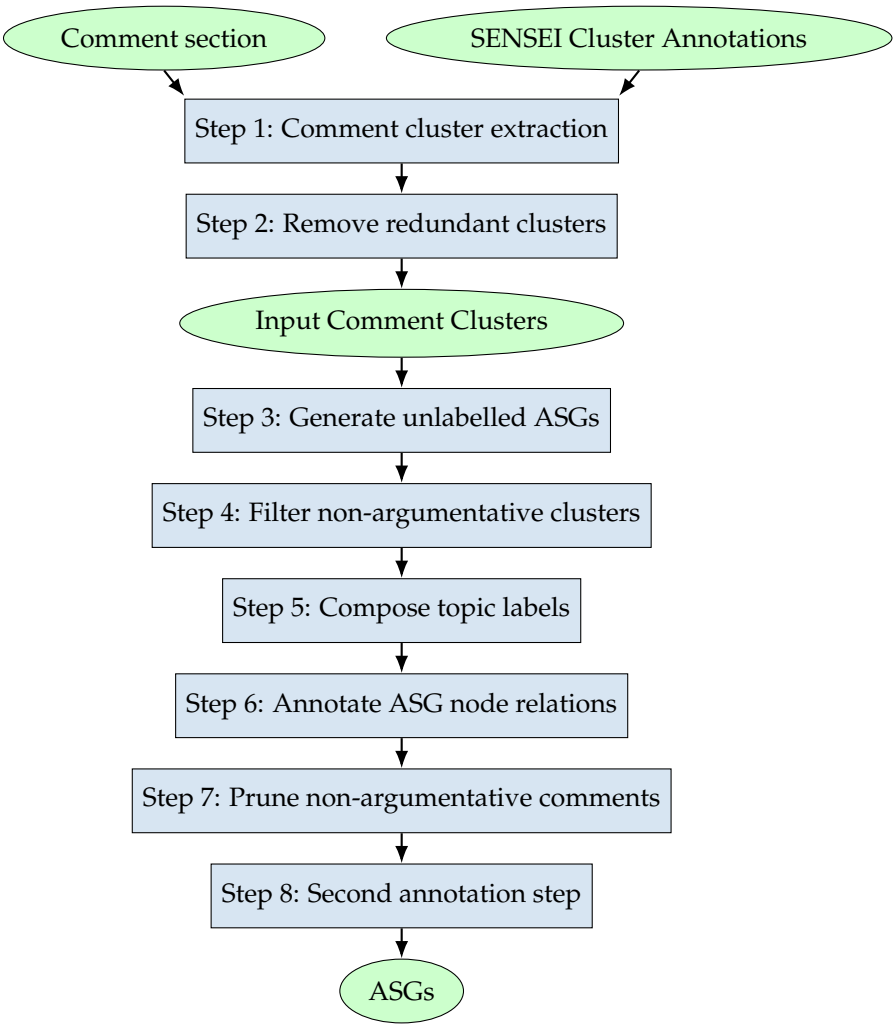


Figure 8.4: The corpus creation process summarised

Step 1: Comment Cluster Extraction

The first step is to extract comment clusters from each comment section. As stated above, the comment sections contain around 100 comments each. Not all of the discussion in a comment section necessarily revolves around the same topic; for example, in the comments responding to the article entitled "Supercarrier made in Britain hailed as flagship for Better Together campaign", some users debated the utility of the aircraft carrier, while others debated Scottish independence.

Algorithm 8.1: *The process for generating an unlabelled ASG, using the category labels and comments available in SENSEI*

1. **Input:**
   (a) A comment section represented as a tree.
   (b) A set of comment identifiers forming a comment cluster on a given topic.
2. Construct a new Argument Structure Graph (ASG) consisting of a single root node labelled *Main topic*.
3. For each comment identifier in the comment cluster:
   (a) If the comment is not already included in the ASG:
      i. Extract the subtree of the comment section rooted at that comment.
      ii. Attach this subtree as a direct child of the *Main topic* node in the ASG.
4. Return the resulting ASG.

In order to extract different debate topics, we made use of the existing GROUP and SUBGROUP clusters in Debatabase. These have some utility for our purposes, because although they are not labelled with an argumentative proposition to debate like the debatabase data, topics under discussion tend to be separate; for example, comments around the utility of the ship were sorted into the topic "The HMS Queen Elizabeth is a waste of money" and those around independence were sorted into the GROUP "Anti-Salmond/ SNP".

Therefore, our first step was to separate the comments into the GROUPS provided by the annotators. However, some GROUPS/SUBGROUPS only included comments from one side of a particular debate (as is suggested by some of the labels, e.g. "Anti-Salmond"). Therefore, when sorting the comments into different arguments, we took not only every comment in the topic, but for each comment we took the entire subtree consisting of this comment and its descendants in the comment section, as shown in Algorithm 8.1. This is because we expect that the arguments against each comment will appear as their children in the original comment structure.

Note that by taking the structure of the argument graph from the reply structure in the original comment section, we make the subsequent step of labelling the attack support relations much less costly: where n is the number of nodes, the number of labels we must generate is equal to $n - 1$ instead of n^2 as it would it would be if we labelled every possible pair of nodes.

Step 2: Remove Redundant Clusters

Due to the fact that there we inserted additional comments to the existing clusters in Step 1, some comments may appear in multiple clusters; in other words, clusters overlap. To avoid redundancy in the dataset, we removed clusters which overlapped excessively.

Our procedure involved automatically examining all pairs of clusters. Taking a pair of clusters as two sets of comments, A and B, if $|A \cap B| \geq 0.6 \cdot |A \cup B|$, we removed the smaller of the two clusters from the dataset.

The output of this step is a set of comment clusters. Each of these clusters is a single "input" for our new dataset. The remaining steps will describe how each of these clusters is used to generate an ASG, our dataset's outputs.

Step 3: Generate an unlabelled argument structure graph

The next step is to generate an unlabelled argument structure graph for each comment cluster. Since, as mentioned, the comments on the Guardian website are tree-structured, with a comment's parent being the comment that it replies to, we can represent each comment cluster as a set of n subtrees.

We then combine all n subtrees to form a single tree, as shown in Figure 8.5. We do this by making M, the cluster topic, the root of the tree, and attaching the roots of all subtrees to M. Note that M is simply the topic label acquired from the original SENSEI dataset. Many of these are not suitable for an ASG, since we want the root to be an argumentative proposition which other nodes attack or support. Therefore, in a later step (**Step 5**), we compose our own topic labels.

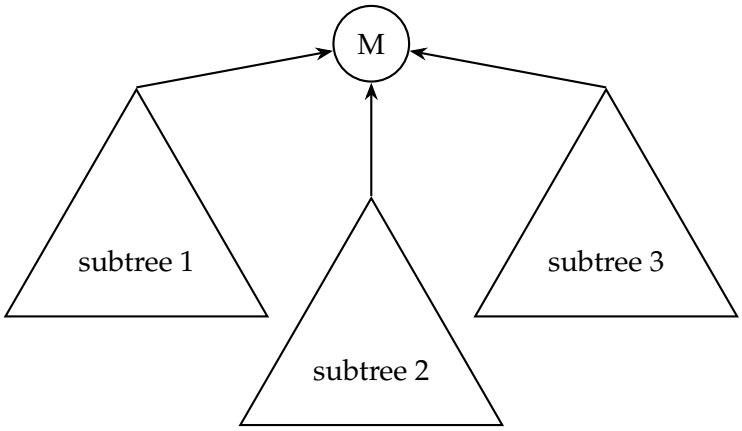


Figure 8.5: An unlabelled argument structure tree

Step 4: Filter Non-Argumentative Trees

Some of the "ASGs" gathered by the previous steps do not actually contain argument, so another stage of filtering is used to remove these. We define "argumentative" as follows: An argumentative comment must do either of the

following:

1. Put across a controversial claim.
2. Attack or support a controversial claim made by another user.

In order to remove non-argumentative "ASGs", we manually labelled them using the following procedure:

1. Examine every level 1 node in the tree and label it as argumentative/ non-argumentative
2. If 50% or more level-one nodes are argumentative, the tree as a whole is labelled as argumentative, otherwise, it is labelled as non-argumentative, and removed from the dataset.

Step 5: Manually Create Argumentative Topic Labels ("Main topics")

This step involves changing the topic labels from the SENSEI corpus, which are somewhat unhelpful for the ASG task, as they typically just consist of a word or noun phrase describing the general subject of the comments, rather than a specific argumentative proposition that can be argued for or against. These labels will become the "main topics" of our ASGs.

These labels were created by a single expert annotator. In order to create the topic labels, the annotator took all the level-1 comments from the ASG, and then attempted to formulate a controversial topic, to which the majority (or all if possible) of these comments appeared to be responding.

Table 8.1 shows an example of depth-1 comments and a manually generated argumentative topic label. In this example, it was possible to write

Depth-1 Comments
Comment 1: There is no need for warships, they are outdated
Comment 2: Ships are waste of money - they will need renewing
Comment 3: Warships act as an effective deterrent
Argumentative Topic Label:
Building a new warship is a waste of money.

Table 8.1: Depth-1 Comments and manually annotated topic label from SENSEI

a label such that all the depth-1 comments can be interpreted as a relevant response. Where not all the Depth-1 comments have an obvious connection, the annotator was instructed to compose a topic label that is relevant to as many of the Depth-1 comments as possible. Any Depth-1 comments that were labelled as irrelevant in the next step were subsequently pruned out.

Step 6: Label Graph Edges

The next step is to label every edge within the output graph. This was done manually by a single expert annotator. The software used to carry out the annotation was a simple Python script which prompted the annotator with both the full text of each comment, and the indicative summary of each comment created by the original annotators of SENSEI.

Figure 8.6 illustrates the interface that the annotator used in order to enter the labels. For each label, our annotator had to press a key to choose between one of three options: “attack”, “support” or “non-argumentative”.

For each pair of comments, the child comment must be labelled as either “attacking”, “supporting” or a third category, “n/a”, which includes anything falling outside these two categories. The full annotation guidelines that we

Discussion Topic: NHS tax is floated by Liberal Democrats to fill £30bn hole

Comment A Summary: Rich - should pay more tax. Low paid workers constantly fear losing home/ assets etc.
Comment A: Great, let the proles squabble amongst themselves which of their number should consider themselves 'rich' (clue anyone who is 'a couple of paychecks away from the street' as they would say in the US, ain't rich), whilst the oligarchs get on with the job of chiselling Olympic sized swimming pools and full-sized ballrooms underneath their London pads and furnishing their yachts with musical waterfalls and flocks of scented sheep and the like. All hail the 'wealth creators' and God forbid that they should pay a fair share of tax, that way lies disaster.

Comment B Summary: Businesses - should pay triple NI tax - people should pay lower income tax
Comment B: Triple the NI for businesses, and lower the income tax. The NI is one tax multinationals can't avoid paying.

Comment B is a reply to Comment A. What is the stance of Comment B towards Comment A? [Attack/ Support/ Neutral]

Figure 8.6: An example prompt presented to the annotator

gave to our annotator are shown in Appendix B.

Step 7: Add Labels and Prune Trees

After the manual annotation has been completed, we then added the gathered labels to the trees. In order to do this, we iterated over all edges in the tree, depth-first. For each edge that had been annotated with "Support", or "Attack", we added the label to that edge. However, for every edge annotated with "NA", we pruned the child node of that edge from the tree, therefore also

pruning all descendants of the child node.

Step 8: Re-annotate removed comments

Algorithm 8.2: *The procedure for re-annotating nodes which were removed in Step 7*

1. **Input:**
   (a) The current Argument Structure Graph (ASG).
   (b) A set of nodes that were pruned in Step 7.
   (c) A set of parent–child relations between the pruned nodes, represented as triples of the form (parent, child, relation).
2. Let the *main topic* be the root node of the current ASG.
3. From the pruned nodes and their relations, construct the maximal forest of pairwise disjoint trees induced by these relations.
4. Sort the resulting trees in descending order of size (measured by number of nodes).
5. For each tree in this ordered list:
   (a) If any node in the tree already appears in the ASG, skip this tree.
   (b) Otherwise, identify the root node of the tree.
   (c) Ask the human annotator to determine whether the root node supports or attacks the main topic.
   (d) If the annotator assigns a *support* or *attack* relation:
      i. Attach the entire tree to the ASG as a subtree of the main topic.
      ii. Label the edge between the main topic and the subtree root with the assigned relation.
6. Return the updated ASG.

In the previous step, when a node y in an ASG was labelled as "N/A" with respect to its parent node, the subtree consisting of y and all of its descendants was pruned from the tree. However, it may be the case that despite being irrelevant to its parent, y or one of its descendants may nonetheless be relevant

to the main discussion topic. We rectify this by obtaining additional human annotations for these descendant nodes; Algorithm 8.2 explains our procedure for collecting these annotations.

8.5 Inter-Annotator Agreement

As discussed above, a second annotator annotated a sample of the labels, in order to validate our annotation guidelines. This sample consisted of 20 comment clusters (out of 70 total), or 282 comment pairs.

		Annotator B		
		Support	**Attack**	**N/A**
Annotator A	**Support**	59	13	7
	Attack	1	116	6
	N/A	32	19	29

Table 8.2: Confusion matrix between Annotator A (rows) and Annotator B (columns).

A Cohen's Kappa of $\kappa = 0.57$ was achieved, indicating moderate agreement. Table 8.2 shows the confusion matrix for the two annotators' labels. The two annotators generally agreed on distinguishing Support from Attack labels, with 59 and 116 instances of agreement for Support and Attack, respectively. The primary disagreements arose when differentiating between argumentative labels (Support/Attack) and non-argumentative segments (N/A), particularly for segments labelled as N/A by one annotator but as Support by the other (32 instances) or as Attack (19 instances). This may reflect an inherent difficulty in identifying whether a comment is relevant to another, something that often depends on inherently ambiguous pragmatic assumptions.

8.6 Description of the SENSEI-ASG Corpus

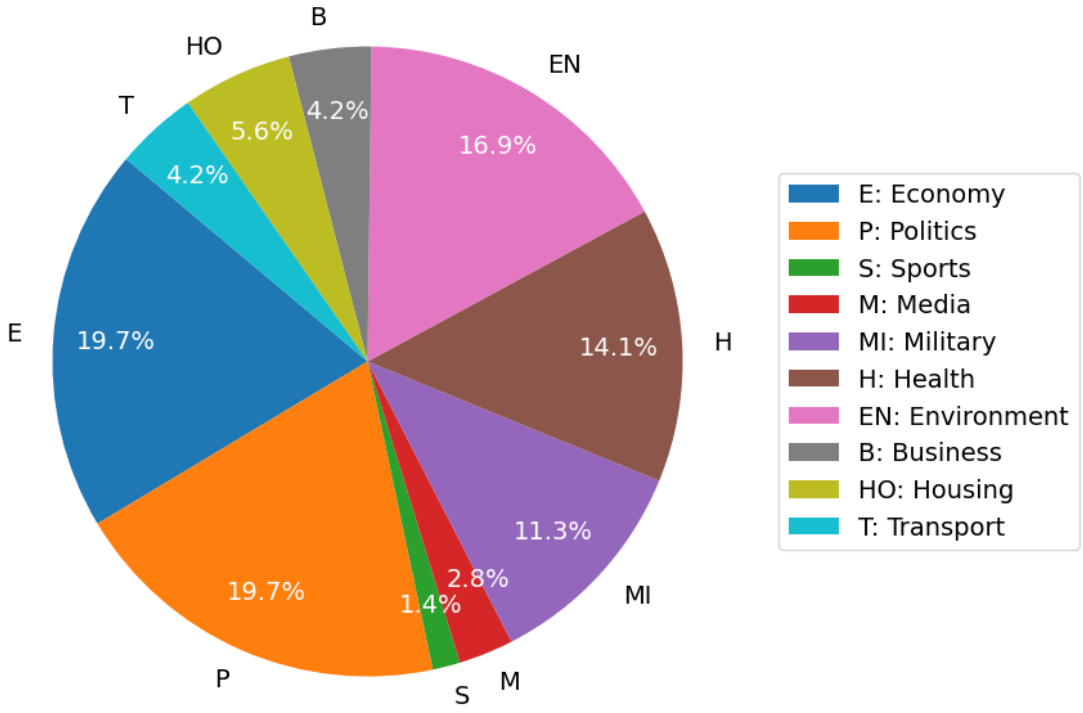


Figure 8.7: The distribution of topic categories in the SENSEI-ASG dataset

In this section, we describe the final corpus resulting from our annotation process. Figure 8.7 gives a breakdown of the topic areas which appear in the final dataset, and Table 8.3 gives an example of an individual Main Topic from each of these areas. Below, we move on to detailing some statistics of the corpus (Section 8.6.1), and then finally describing the two variants of the corpus which we have produced to address different research questions (Section 8.6.2).

Category	Example topic
Economy	A tax hike would damage businesses and the economy as a whole
Politics	The police and government should not have access to people's personal data
Military	The HMS Queen Elizabeth is a waste of money
Environment	Climate change is man made
Health	The British government should have reacted more quickly to the Ebola epidemic
Business	Tesco pays low wages
Housing	New build houses are either too expensive or too small
Transport	Network Rail is to blame for increased train ticket prices
Media	This newspaper article is bad journalism
Sports	The English football team played poorly

Table 8.3: Examples of "main topics" from each category shown in Figure 8.7

8.6.1 Corpus Statistics

Table 8.4 contains some statistics describing our generated dataset. Unlike the Debatabase-ASG data, which was artificially constrained to contain a roughly equal number of support and attack relations, it can be seen that our dataset is heavily skewed towards attacks (72.5% of the labeled relations), which reveals that commenters on the Guardian website tend to attack other comments far more frequently than they support them.

In contrast to Debatabase-ASG, which contains only trees of depth 2, the average ASG in SENSEI-ASG has a depth of 5.7. The deepest has a depth of 12. Table 8.5 provides an example of an input cluster and the corresponding ASG, both in text format and in tree format, and Figure 8.8 shows a graphical plot of the corresponding ASG.

N. Comment Cluster/ ASG Pairs	70
Train: Val: Test split	57 : 7 : 6
Average Input Cluster Length (overall words)	1011
Average ASG length (overall words)	343
Average comments per Cluster	16.4
Average tree depth	5.7
Max tree depth	12
Min tree depth	2
N. Attack relations	797
N. Support relations	302

Table 8.4: SENSEI-ASG Corpus Statistics

8.6.2 SENSEI-ASG Variants

We have created two variants of the corpus, to address different research questions relating to ASGP. These correspond to the two different variants of the AGSP task defined in Chapter 5, Section 5.3.5.

SENSEI-ASG-Rep: This is the main variant of the corpus, which is more suitable for research that focuses on applying ASGP in a realistic setting. This corpus supports the overall task pipeline as presented in Figure 8.3. It contains reply information, i.e. information about which comment replied to which other comment in the original comment section. Furthermore, unlike the other variant, we include some irrelevant and non-argumentative comments in the input, which the model must ignore when creating an ASG.

SENSEI-ASG-NoRep: This variant was differs from the main SENSEI-ASG-Rep variant in two ways: firstly, no reply information is contained in the input, and secondly, all comments which appear in the input also have summaries in the output.

We have produced this variant using one extra simple post-processing step: we removed all comments which lacked a corresponding summary from the comment clusters that were produced in Step 2 of the pipeline process described in Section 8.4. In the experiments in Chapter 9, we will use this variant instead of SENSEI-ASG-Rep, because we want to test the transferability of the Debatabase-ASG data, which is formatted identically to this variant.

8.7 Conclusion

In this chapter, we have described our procedure for creating a dataset, SENSEI-ASG, using a subset of the data from the SENSEI corpus (Barker et al., 2016). This dataset is unique in that it contains Argument Summary Graphs, formatted as per our schema in Chapter 5, annotated from naturalistic dialogue (namely, comments on an online news website). The dataset that we generate is smaller than our previously generated Debatabase-ASG corpus (Chapter 6), but contains trees which are significantly larger and more complex than those which appear in that corpus.

8.8 Limitations and Future Work

The Dataset that we have generated in this chapter was annotated by a single expert annotator. There is therefore no measure of inter-annotator agreement which could be used to evaluate the extent to which other annotators agree

with the decisions made during the annotation process. In future work, it would be extremely valuable to repeat the corpus annotation processllama 2 with several expert annotators in order to obtain a more reliable dataset.

Secondly, in this dataset, we have only considered inter-comment relations (attacks and support between comments), instead of inter-ADU relations. The reason for this is that it greatly simplifies the annotation process, as we remove the necessity to annotate ADU boundaries, and additionally greatly reduce the number of relations which need to be annotated subsequently. However, in some cases, more complex relations may exist between comments (e.g. comments which simultaneously attack and support different parts of a parent comment), and it may be worthwhile exploring the possibility of elaborating the annotation system to capture these complexities.

A final limitation is scale; although there are 1148 comments in this dataset in total, which is comparable to the size of other datasets within the field of Argument Mining, it is a relatively small dataset within the realm of machine learning, where collecting vast amounts of data has been shown to be the most reliable way to achieve good results. For this reason, future work should focus on techniques for generating larger scale training corpora for this task.

One potential avenue for this is the concept of developing a “game with the purpose” for the goal of Argument Summary Graph generation. “Games with a purpose” are a form of crowdsourcing, wherein useful annotations are generated by online participants playing a game involving extracts of the corpus to be annotated Von Ahn (2006). Investigations into this topic (unpublished) have been carried out with by an undergraduate student, with the supervision of the supervisor of this thesis and the author. This work has proposed several types of “game" that could be investigated in future work, for example:

1. A “classification" game in which a user must decide on the relation be-

tween two text units, e.g. two ADUs or two comments (optionally also rating the "strength" of the relation on a sliding scale)

2. A "debating" game in which a user must formulate their own arguments in response to either an AI agent or another user.

3. A "span labelling" game in which highlighted segments of a text must be categorised according to ADU type.

Although it may seem intuitively implausible for a large number of participants to spend time on this sort of task without compensation, there are previous examples of such games being successfully employed to generate large datasets for what seem like subjectively dry tasks, such as coreference resolution (Poesio et al., 2013). Additionally, the existence of large collaboratively-created argument resources such as `kialo.com` and the popularity of competitive debate fora such as `reddit.com/r/changemyview` both indicate that debate holds intrinsic interest for a substantial number of people.

Table 8.5: Example inputs and outputs from SENSEI-ASG Dataset

Example Input
Main topic: South-Eastern England gets disproportionately high levels of public funding.
Comment 1 (reply to Main topic): The most heavily subsidised rail users in the region getting nearest the targets are given yet more disproportionate subsidy. Anyone spot any vote buying going on here? Cameron and Osborne have just reneged on their pledges to fund electrification in South Wales. This is just more greed and selfishness from the South East, the tape worm of the UK public sector economy.
Comment 2 (reply to Comment 1): It's the same with all government funding - be it transport, arts, or whatever. London and its surrounds always get a disproportionately large slice of the pie!
Comment 3 (reply to Comment 2): Are you sure that per capita London is better funded than other parts of the country? I have my doubts.
Comment 4 (reply to Comment 3): Yes, just do some searching on the web. There's plenty of information out there!
Comment 5 (reply to Comment 1): Right, take money from all rail travellers and give it to the SE commuters.
Comment 6 (reply to Main topic): How will an improved Wifi system speed up train journeys? A: By distracting passengers for longer periods so they don't notice how late their train is running. How will this benefit West country users?...
Comment 7 (reply to Comment 6): When you're late and stuck on open line somewhere, you can carry on sending emails and tele-working from the comfort* of your seat**. * we should stress that comfort is a relative concept (rather like punctuality) and that your seat may not actually be all that comfortable, especially if you are more than 2 feet wide. ** may not be available. May not be comfortable if available (see above). In which case you have no discomfort about which to complain!
Comment 8 (reply to Comment 6): Why do you seek to isolate the South East Viper? It has the busiest stations and some of the most crowded trains on the network.
Example Output
Comment 1 (supports Main topic): se getting disproportionate public funding; vote buying
Comment 2 (supports Comment 1): london getting disproportionate public funding
Comment 3 (attacks Comment 2): questions london getting disproportionate public funding
Comment 4 (attacks Comment 3): london getting disproportionate public funding -agrees– look on web for article
Comment 5 (supports Comment 1): se getting disproportionate public funding
Comment 6 (supports Main topic): privatised essential industries = privateering corporations; ””privateer = government authorised pirate””
Comment 7 (supports Comment 6): wifi on trains– how can you use it when no room and no seats due to crowding; humour
Comment 8 (attacks Comment 6): defending s-east–is very crowded and busy

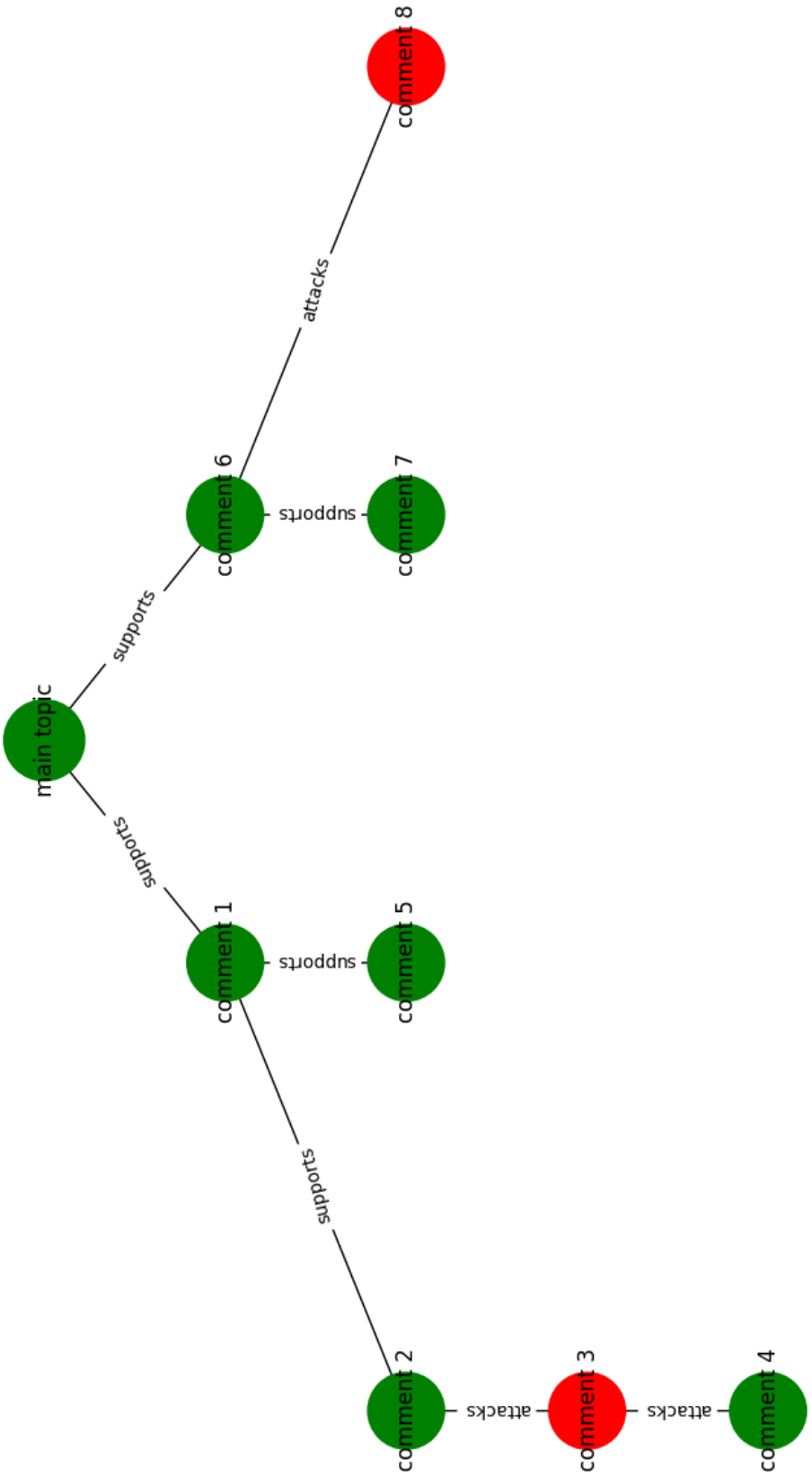


Figure 8.8: The Argument Summary Tree in Table 8.5 visualised as a graph.

Chapter 9

End-to-end Argument Summary Graph Parsing: A Multi-Corpus Evaluation

9.1 Introduction

In this chapter, we evaluate the performance of training end-to-end models across different Argument Mining corpora. Specifically, we look at the two corpora for Argument Summary Graph Parsing (ASGP): Debatabase-ASG and SENSEI-ASG, which we created and introduced in Chapters 6 and 8 respectively. The third corpus used in this study is the Argument Annotated Essays Corpus (Stab and Gurevych, 2014a), a corpus for Argument Structure Parsing (ASP). We investigate whether any performance gains will result on the ASGP task from additionally training on ASP data, and vice versa.

We again apply the end-to-end LLM framework used previously in Chapters 6 and 7, but this time, we fine-tune just one model, LLaMA-3, on different combinations of input datasets, and evaluate it using the test portions of the three different datasets individually.

We choose to use the LLaMA-3 model due to its strong performance on ASP in Chapter 7. In order to confirm that it is suitable for ASGP as well, we

validate its performance on the Debatabase-ASG dataset, comparing it to the other models studied in Chapter 6.

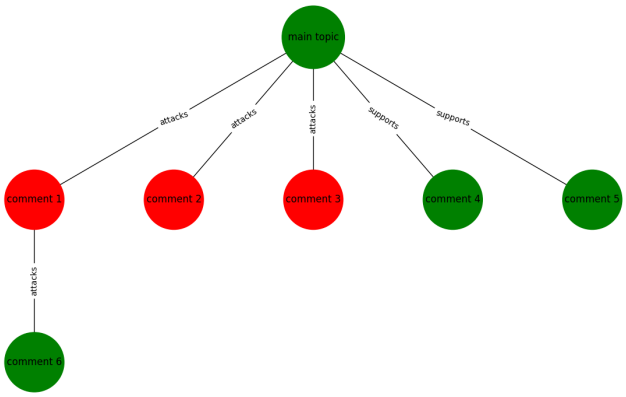
The main contributions of this chapter are as follows:

1. We compare performance of the same model on Debatabase-ASG, and our more-recently created corpus, SENSEI-ASG, and find that the latter is much more challenging, due to the greater complexity of this corpus.
2. We carry out experiments using all possible combinations of three training datasets, including Debatabase-ASG, SENSEI-ASG, and AAEC. We find that training with an additional dataset (whether from the same genre and task, as is the case with Debatabase-ASG, or from a different genre and task, as is the case with AAEC) can provide performance benefits when testing on SENSEI-ASG.
3. We find that the LLaMA-3 model outperforms all models that we have previously tested on ASGP in Chapter 6 on the Debatabase-ASG dataset.

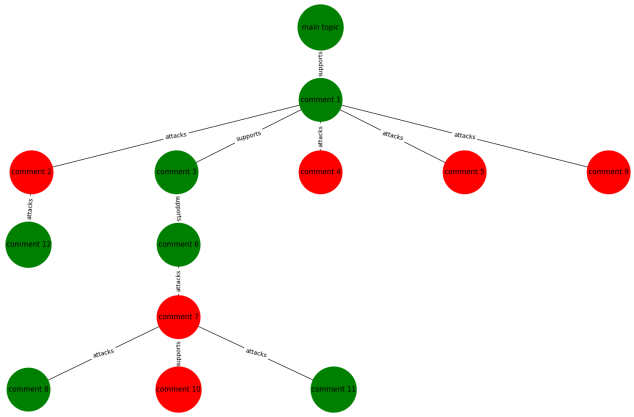
9.2 Background

Two related tasks in Argument Mining are Argument Structure Parsing (ASP) and Argument Summary Graph Parsing (ASGP). Argument Summary Parsing is the task of identifying all Argumentative Discourse Units (ADUs) in a text, classifying them into different ADU types (e.g. Claim and Premise), before identifying the links between them, and classifying those links into different types (e.g. "attack" and "support").

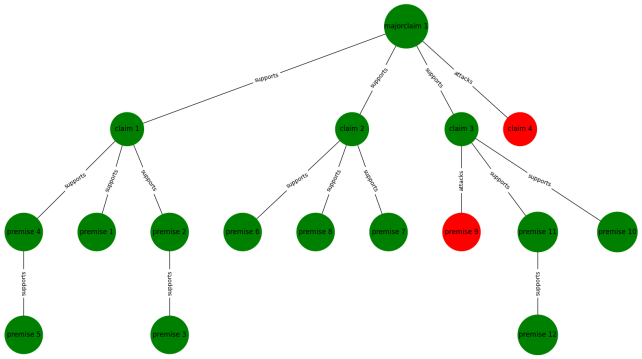
ASGP, as we have defined the task in Chapter 5, also involves building a graphical summary of an argumentative text, namely a dialogic text, such



(a) Debatabase-ASG



(b) SENSEI-ASG



(c) AAEC

Figure 9.1: Example argument trees from each of the three datasets investigated in this chapter. Nodes with a positive stance on the main topic M (the tree root) are shown in green, and those with a negative stance towards it are shown in red.

as an online comment section. In this task, we identify links not between ADUs, but between comments made by different users. These links are also labelled as "attack" or "support" relations. Unlike ASP, however, the ASGP task additionally involves generating a short summary of each comment, such that we are finally left with a tree structure containing summaries of each comment and the relations between them.

A wide variety of approaches from the field of machine learning have been applied to try and solve the ASP task; these are reviewed in Chapter 3, Section 3.4. However, the approach most similar to the one that is employed here is Kawarada et al. (2024), which attempts the task of ASP end-to-end as a text-to-text translation task (i.e. generating a text-based representation of an argument graph from the unprocessed text input using a single Large Language Model fine-tuned for this task).

In Chapter 6, we applied a similar end-to-end technique to the AGSP task on our Debatabase-ASG dataset, and found that this method performed better than the alternative pipeline technique that we used. In Chapter 7, we replicated Kawarada et al. (2024)'s method and found that it performed close to the state of the art for ASP.

Due to the fact that the text-to-text translation framework performs well for both ASP and ASGP, a logical next step is to train a single model for both tasks, which we attempt in this chapter.

9.2.1 Previous Works on Cross-Corpus Evaluation

Several works exist that have performed cross-corpus evaluations within ASP (i.e. training on one or several corpora and evaluating on another). These include Morio et al. (2022), who built an end-to-end neural model, which we

have described in more detail in section Chapter 3, Section 3.4, which they then used to test transferability between corpora by training using an auxiliary corpus, i.e. an additional corpus used for training on top of the target corpus. They evaluate their model on each possible permutation of (target, auxiliary) pairs from five different corpora, including the AAEC and CDCP [17]. They find a high degree of variability in benefit from using an auxiliary corpus, with many resulting in small improvements, while in other cases using the auxiliary corpus was actually detrimental for certain metrics. Some patterns emerge, such as the smaller target corpora, with less training data, (CDCP and MTC) benefitting from an auxiliary corpus much more. The authors suggest that similarity of annotation schemes was also important for transferability.

Gemechu and Reed (2025), whose work we have also reviewed in Chapter 3, Section 3.4, also carry out a cross-corpus evaluation, using one monological (AAEC) and two dialogical corpora using IAT: QT30 and US2016. In this evaluation, they trained on every possible permutation of two models and evaluated on the third. They found that their CU-MAM excelled at cross-corpus evaluation compared to other baselines; however, performance was in all cases lower than in-dataset evaluation.

Our work differs from these two mentioned works in that we investigate transfer not just between monological and dialogical text, but between two separate tasks; ASP and our novel task of ASGP.

[17] See Table 2.1 in Chapter 2 for information on each of the corpora mentioned in this section by its acronym.

9.3 Data

We use three different corpora in this chapter. Two of them are datasets for ASGP which we have developed: Debatabase-ASG (described in Chapter 6) [18] and SENSEI-ASG (described in Chapter 8). The third corpus is the Argument Annotated Essays Corpus or AAEC (Stab and Gurevych, 2014a). This is an ASP corpus in the genre of student essays, which we have described in detail in Chapter 7, Section 7.4.

As we have described in Chapter 8, Section 8.6.2, we have developed two different variants of SENSEI-ASG to address different research questions. The variant of SENSEI-ASG that we use in these experiments is SENSEI-ASG-NoRep — the variant that does not contain information about which comments respond to which other comments in the input. We do this in order to have a fair comparison with the Debatabase-ASG dataset, which also lacks this reply information.

Figure 9.1 shows an example graph from each of the three datasets. We can see that each corpus has a tree structure. In the case of AAEC, which is monological, the root is the Main Claim of the essay, whereas for the Debatabase-ASG and SENSEI-ASG datasets, which are dialogical, the root is the main topic of the discussion. We can also observe several features which are typical of the three datasets. Firstly, the Debatabase-ASG tree is much smaller (both in terms of depth and in terms of the overall number of nodes) and therefore less complex than either of the other datasets. Secondly, we can see from the colour of the nodes that the AAEC tree has a large preponderance of "support" links (as one would expect from a monological persuasive essay) while the other

[18] For the Debatabase-ASG data, we use the "depth-2" data setting described in Chapter 6, which includes trees of up to a depth of 2, unlike the depth-1 setting which only includes depth-1 or "flat" trees

	Corpus		
	AAEC	**SENSEI-ASG**	**Debatabase-ASG**
Task	ASP	ASGP	ASGP
Domain	Student Essays	Online Comment	Online Comment
Size of Dataset (words)	147,271	71,884	632,062
Average Tree Depth	3.56	5.7	2
Average N. Nodes	12.95	16.4	7
Average Comment/ADU Length (words)	20.9	61.6	176
Average Node Summary Length (words)	N/A	20.9	17.8

Table 9.1: Descriptive statistics comparing the three corpora that we experiment with in this chapter.

two are somewhat more balanced between "support" and "attack".

In Table 9.1, we provide some descriptive statistics summarising the three corpora. This table reveals the relative complexity of the three datasets, using multiple different features. As explained in the previous chapter, the SENSEI-ASG dataset is more realistic than the Debatabase-ASG dataset, due to the fact that it was sourced from a set of spontaneous arguments in news website comments sections which have been annotated, rather than a carefully curated collection of online debates. This fact appears reflected in the fact that SENSEI-ASG has both a greater tree depth and a higher average number of nodes per tree.

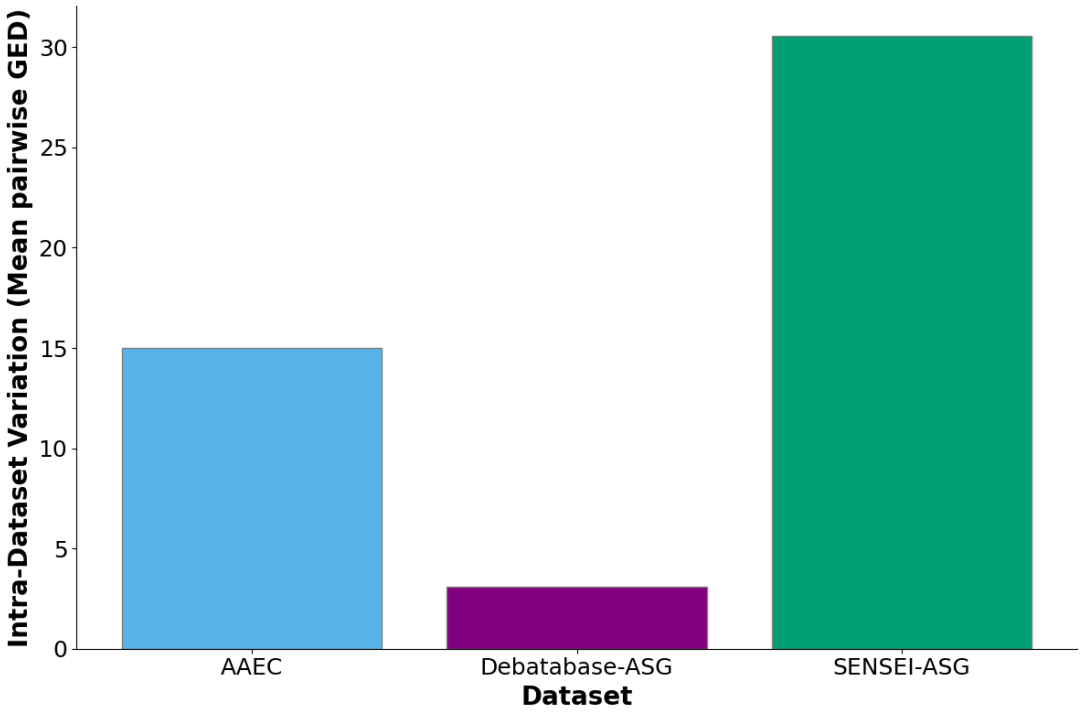


Figure 9.2: The intra-dataset variablity of the three datasets used in this chapter, measured using mean pairwise Graph Edit Distance.

9.3.1 Measuring Dataset Compexity

As another measure of dataset complexity, we looked at the intra-dataset variation in terms of tree structure. Our motivation for looking at this is that it may be hard for a model to learn to output diverse types of tree structure. As a way to measure this diversity, we have used a pairwise Graph Edit Distance metric, which measures the distance, in terms of structure, between the different graphs which make up the dataset. Due to the intractably large number of pairs of graphs in each dataset, we calculated this metric for a sample of 100 pairs for each dataset, and then took the average [19]. The resulting mean intra-dataset GED values are shown in Figure 9.2.

[19] More details of how we calculated the GED in this instance are contained in Appendix A, Section A.4.

These figures, which show rather low variation in the Debatabase-ASG, higher variation in AAEC, and even higher variation in SENSEI-ASG, are unsurprising if we consider the annotation methods for the three datasets. Debatabase-ASG uses short comments sections with exactly 6 comments, and where the maximum tree depth is constrained to be 2 due to the source data, and therefore cannot vary much. The AAEC allows more complex structures, but there were some constraints in the annotation process such as choosing essays of roughly the same length. In creating SENSEI-ASG, however, we did not filter the argument graphs based on size, and therefore much larger or much smaller graphs are possible.

9.3.2 Dataset Formatting

Table 9.2 illustrates how the datasets are formatted for the experiments carried out in this chapter. The output formatting for the ASP dataset (AAEC) is the same as that described in Chapter 7, Section 7.4.2, or, more specifically, the "list formatting" approach that we describe in that chapter. The formatting for both ASGP datasets (SENSEI-ASG and Debatabase-ASG) is the same as that already described in Chapter 6, Section 6.4.3.

The inputs are also formatted as stated in the sections linked above; for the AAEC, the input is simply the original essay text, without any pre-processing, and for the ASP corpora, the input is the "main topic" of an argument, followed by a list of numbered comments.

As we can see, there are minor differences between the formatting for the two tasks. Firstly, when it comes to ASP, the MajorClaim is not given in the input, since part of the task is to extract this from the essay; conversely, for the ASGP datasets, the Main Topic does appear in the input, since in our framing of the task we assume that a model will not be able to infer it.

AAEC	MajorClaim 1: we should do our parts to do whatever it takes to protect old buildings, letting next generations still have a chance to look at them. Claim 1 (supports MajorClaim 1): Historic buildings have intrinsic values. Premise 1 (supports Claim 1): they are significant symbols of a city.
SENSEI-ASG	Comment 1 (supports Main topic): se getting disproportionate public funding; vote buying Comment 2 (supports Comment 1): london getting disproportionate public funding ? Comment 3 (attacks Comment 2): questions london getting disproportionate public funding

Table 9.2: Comparison of the output formatting for an ASP corpus (AAEC) and an ASGP corpus (SENSEI-ASG).

The second, minor, difference is that the ASGP output contains "Comment IDs" while the ASP output contains "ADU ids" that encode the ADU type as well as a number. However, as we will discuss further in Section 9.4, we believe that these two text formats may be similar enough that fine-tuning on one may be helpful for another.

9.4 Research Questions

In the following experiments, we address 3 main research questions, which we enumerate below.

RQ 9.1 — How does model performance compare across the two different ASGP test sets, Debatabase-ASG and SENSEI-ASG? Our first research question is to compare the performance of identical models across these two datasets. In doing so, we aim to verify whether the higher dataset complexity of SENSEI-ASG, described in the previous section, makes the task more challenging, and if so, to what extent.

RQ 9.2 — What are the effects of fine-tuning on different combinations of datasets on task performance? Secondly, we investigate the effects on performance on SENSEI-ASG when we use additional fine-tuning data. We investigate how much performance can be improved by using both Debatabase-ASG, a dataset for the same task (ASGP) in the same text genre (online debate) and the AAEC, a dataset for a distinct but related task (ASP) in a different genre (argumentative essays).

RQ 9.3 — How does LLaMA-3 compare to the LLMs we have previously evaluated on this task? We use a state-of-the-art LLM, LLaMA-3 (Grattafiori et al., 2024), for all of the experiments carried out in this chapter. Our final research question is to test whether this can perform better than the other models which we have used for ASGP in Chapter 6, by evaluating its performance on the Debatabase-ASG dataset.

9.5 Evaluation

In order to evaluate this task, we use three metrics, which we have described more fully in Chapter 5, Section 5.4, to evaluate three different aspects of the system output. These are:

1. **ROUGE score**: (Evaluating the quality of output summary text)
2. **Node Position F1** (Evaluating the correctness of the graph structures generated by the model)
3. **Node Stance F1** (Evaluating the correctness of the attack/ support relations predicted by the model)

Thse are the same metrics which we applied to evaluate the Debatabase-ASG dataset in Chapter 6, with the exception that we have chosen to use Node Position Accuracy to evaluate the correctness of the graph structure, instead of Graph Edit Distance (GED). While GED and Node Position Accuracy evaluate the same information, we choose to use the latter here rather than the former because it is more easily interpretable in the context of our work. Firstly, it increases, rather than decreases, with the correctness of the graph structure, bringing it in line with the other metrics we use, and secondly, as with the Node-stance-F1 metric, a perfect result with Node Position Accuracy gets a score of 1, making the interpretation of this statistic more intuitive compared to GED.

For the sake of consistency, we evaluate the AAEC dataset using these metrics, instead of the typical Component-F1 and Relation-F1 metrics which are used in ASP (which we use for evaluating models on AAEC in Chapter 7).

9.6 Methodology

We carried out fine-tuning using the three different test sets, SENSEI-ASG (S), Debatabase-ASG (D), and Argessays (A), and all seven different possible combinations of training sets, which we will denote {S, D, A, S+D, S+A, D+A, S+D+A}. We carried out a separate hyperparameter tuning run using grid

search for each training set/ test set pair, using the appropriate validation data for each target test set; hence, we carried out a total of 21 grid search runs (3 test sets × 7 training set combinations). Inputs and outputs for all experiments are formatted as described in Section 9.3.2.

We chose to train the LLaMA-3-8B model (Grattafiori et al., 2024) for this task, due to its high performance on the ASP task that we explored in Chapter 7. Fine-tuning was conducted on a high-performance computing (HPC) node consisting of 8 NVIDIA V100 GPUs. Due to the large size of the LLaMA-3-8B model, we used QLORA (Dettmers et al., 2023), a parameter-efficient fine-tuning technique (explained in Chapter 2, Section 2.6).

We fine-tuned for 8 epochs, using the AdamW optimiser (Loshchilov and Hutter, 2017). Early stopping was employed with a patience parameter set to 2 epochs, i.e. training was halted if the validation performance did not improve for two consecutive epochs. The hyperparameters tuned were learning rate (from the set $\{1e{-}4, 1e{-}5\}$), and weight decay (from the set $\{1e{-}2, 1e{-}3, 1e{-}4\}$). The other hyperparameters were set to the values shown in Appendix A.3.

9.7 Results

Tables 9.3 – 9.5 show the scores achieved by each model across each training-set / test-set pair for the three evaluation metrics. The colour coding in the cells indicates performance, with purple corresponding to higher scores and white to lower scores on each metric. Below we will analyse the results obtained with respect to each of the research questions stated in Section 9.4.

		Training Set						
		S	D	A	S+D	S+A	D+A	S+D+A
Test Set	S	0.181	0.206	0.117	0.229	0.228	0.242	0.245
	D	0.169	0.577	0.059	0.566	0.157	0.581	0.535
	A	0.296	0.121	0.848	0.240	0.862	0.872	0.853

Table 9.3: ROUGE scores (LLaMA-3)

		Training Set						
		S	D	A	S+D	S+A	D+A	S+D+A
Test Set	S	44.30	63.40	0.00	56.90	45.00	57.10	49.14
	D	35.86	92.09	0.00	91.48	50.48	94.63	95.49
	A	0.00	0.00	49.72	0.00	50.38	51.28	54.98

Table 9.4: Node Stance F1 (LLaMA-3)

		Training Set						
		S	D	A	S+D	S+A	D+A	S+D+A
Test Set	S	55.12	40.24	8.82	55.28	53.35	23.87	49.65
	D	56.19	90.48	14.29	93.10	69.29	91.43	92.14
	A	8.30	8.30	43.36	8.30	48.13	46.12	45.93

Table 9.5: Node Position F1 (LLaMA-3)

9.7.1 Model Performance on Debatabase-ASG vs SENSEI-ASG (RQ 9.1)

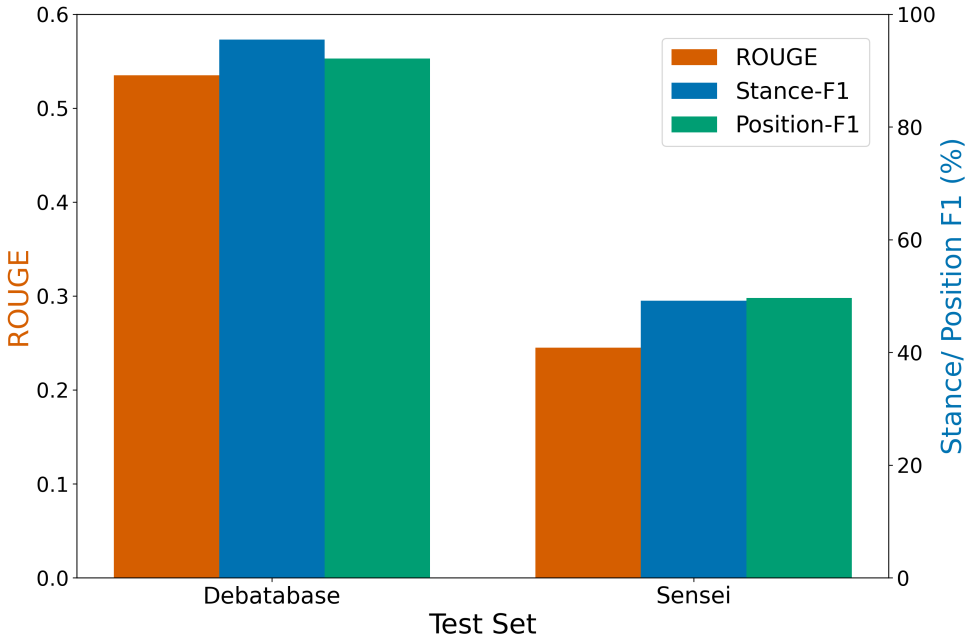


Figure 9.3: Performance of the best model (LLaMA-3 S+D+A) compared on the Debatabase-ASG and SENSEI-ASG test sets

Figure 9.3 compares the performance of fine-tuned LLaMA-3 on two different test sets: Debatabase-ASG and SENSEI-ASG. We compare the model which performed best on each tests set, which in both cases happened to be the model fine-tuned on all three datasets (the S+D+A training set).

The comparison shows that SENSEI-ASG is much more challenging than Debatabase-ASG, with the model achieving around half the performance on the former dataset compared to the latter.

9.7.2 Fine-tuning with Additional Datasets (RQ 9.2)

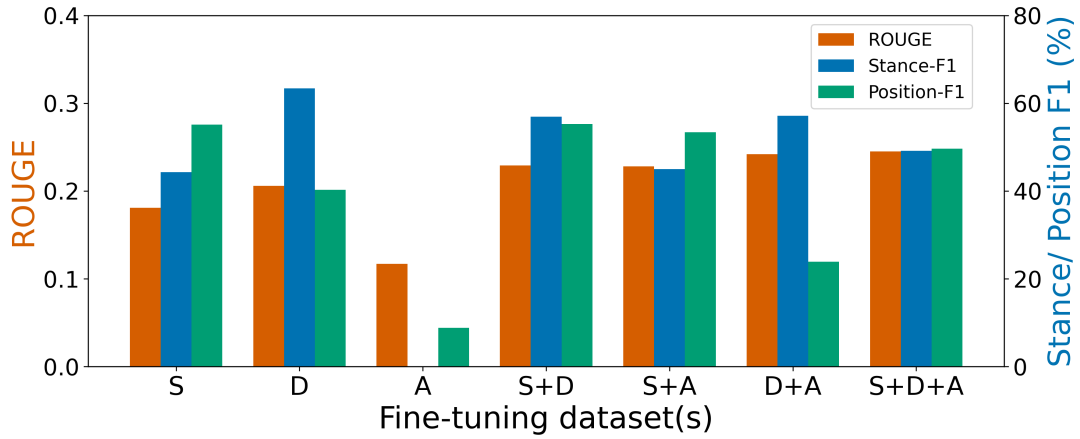


Figure 9.4: Performance of the fine-tuned LLaMA-3 model on the SENSEI dataset with different fine-tuning sets

Figure 9.4 shows the performance of all the permutations of datasets on SENSEI-ASG. Overall, when we consider those models that performed well across the three metrics, it would appear that the best performing combinations of training sets are S+D, S+A and S+D+A; these are the combinations which contain both in-dataset data (SENSEI) and (an) additional training set(s).

One interesting observation is that performance does not seem to improve when a third dataset is added (S+D+A) compared to either S+D or S+A. This could reveal a limit to performance gains which can be achieved by adding training data that differs qualitatively from SENSEI-ASG. In other words, perhaps a greater quantity of highly *relevant* data must be added in order to increase performance further.

9.7.3 LLaMA-3 Performance on Debatabase-ASG (RQ 9.3)

Model	ROUGE ↑	Stance-F1 ↑	GED ↓
Results from Chapter 6 (all trained on Debatabase only)			
Phi-2	0.5460	0.7825	2.5667
LLaMA 2	**0.5829**	0.8700	1.45
GPT-3.5	0.5422	0.8333	1.85
Experiments in this chapter			
LLaMA-3-D	0.577	*0.921*	*1.21*
LLaMA-3-S+D+A	0.535	**0.955**	**0.85**

Table 9.6: End-to-end (fine-tuned) results on Debatabase-ASG. Higher ROUGE-2 and Stance-F1 are better (↑), lower GED is better (↓). GED is used here to allow comparison with the work carried out in Chapter 6. The best-performing results are shown in **bold**, and the second-best in *italics*.

Table 9.6 compares the fine-tuned LLaMA-3 model used in this chapter to the best-performing models from Chapter 6. Since all of the experiments in that chapter involved models trained only on the Debatabase data, we have provided a comparison to the LLaMA-3 model trained on Debatabase only (LLaMA-3-D). We have found this outperformed all other models in two out of three metrics, confirming that LLaMA-3 is a suitable model for this task. We also show results for the LLaMA-3 model fine-tuned on all three datasets, which obtains state-of-the-art results on two out of three metrics.

9.8 Conclusion

In conclusion, we have confirmed our intuition that SENSEI-ASG is a more challenging dataset than Debatabase-ASG. We have found that fine-tuning with additional data on top of SENSEI-ASG, either an out-of-genre ASP dataset (the AAEC) or an in-genre ASGP dataset (Debatabase-ASG) can improve results on SENSEI-ASG. However, fine-tuning using *both* additional datasets does not appear to give an added benefit above using one or the other. This suggests that in order to derive further performance benefits on SENSEI-ASG, what is needed may be a greater quantity of *relevant* data, i.e. additional data containing comments and ASGs with a similar degree of complexity to that found in SENSEI-ASG.

9.9 Limitations and Future Work

One limitation is that we used only one language model, LLaMA-3, to carry out the experiments described in this Chapter. The conclusions about transferability between the two tasks are therefore somewhat tentative, and it would be productive in the future to carry out experiments with multiple LLMs, as well as looking at ASP datasets other than the AAEC and evaluating how well models trained on these datasets can perform on the ASGP task.

Chapter 10

Conclusion

In this thesis, we have addressed an important and understudied area of Argument Mining: the use of graphical structures in the summarisation of argument.

We have followed the intuition, laid out in Chapter 1, that graphical summaries are well suited to deal with various properties of argumentative dialogue. For example online argument often has a “nested" structure, wherein arguments can develop around the premises of another argument, and may contain redundant comments.

We have specifically proposed a type of structure called an Argument Summary Graph or ASG, which we flesh out in Chapter 5. ASGs are tree-structured, with nodes which represent summaries of individual arguments and edges which represent the argumentative relations between these summaries. The largest contributions of this thesis have been to develop resources and models to automatically generate these structures.

In the sections below, we will first explain how our research ties in to the Research Goals mentioned in Chapter 1 (Section 10.1) before moving on to summarise our various contributions: the tasks we have proposed in the thesis (Section 10.2), the data resources created (Section 10.3), and the experimental results obtained (Section 10.4). Finally, we propose future research directions

based on this project (Section 10.5).

10.1 Research Goals

In Chapter 1, Section 1.2, we proposed three research goals, which we repeat here:

Research Goal 1: To investigate the performance of different Natural Language Processing techniques, in particular Large Language Models, in producing graphical summaries of dialogue.

Research Goal 2: To develop resources for the generation of graphical summaries of argument.

Research Goal 3: To investigate techniques for the generation of free-text summaries from graphical summaries.

Our contributions towards **RG1** include our proposal of Argument Summary Graph Parsing (Chapter 5), and the experimental work done on the ASGP datasets that we created (Chapter 6, Chapter 9). The work in Chapter 7, on Argument Structure Parsing, was not directly relevant to this, or any of the research goals, but arguably adjacent due to our applying the same type of end-to-end techniques that we have used for ASGP to ASP.

The work carried out towards **RG2** is the development of our two ASGP datasets, Debatabase-ASG (Chapter 6) and SENSEI-ASG (Chapter 8).

Finally, we addressed **RG3** in Chapter 4, in our work on Reasoning Marker prediction. This task is intended to be part of a pipeline for generating free-text summaries from ASGs.

10.2 Proposed Tasks

We have proposed two separate tasks in this thesis:

Argument Summary Graph Parsing: (Chapter 5) A major contribution of this thesis is the proposal of Argument Summary Graph Parsing (ASGP). This is a novel kind of argument summarisation, which involves providing succinct summaries of all comments in a dialogic text and identifying the argumentative relations between them. In developing this task, which has a different focus from existing research in the AM field, we hope to stimulate further work on extracting these kinds of graphical summaries, which are useful for aiding reader understanding of argumentative dialogues.

Reasoning Marker Detection: (Chapter 4) Another contribution is our proposed task of Reasoning Marker Detection, which we developed as part of a proposed pipeline for converting graphical summaries into text-based summaries. This is a binary classification task which involves predicting whether a Reasoning Marker is present or absent based on the surrounding text.

10.3 Data Resources

Three data resources were developed over the course of this PhD project:

Reasoning Marker Detection Dataset: A dataset for the task of Reasoning Marker Detection, which is a preprocessed version of the Argument Annotated Essays Dataset (AAEC) (Stab and Gurevych, 2014a).

Debatabase-ASG: A dataset for the task of Argument Summary Graph Parsing (ASGP). It has two separate modes - depth-1 and depth-2, in which the ASG is either a flat tree with all nodes linked to the root, or a depth-2 tree where nodes can also attack/ support higher level nodes. The data is sourced from idebate.net, a collection of pro and con arguments on a variety of topics curated by the International Debate Education Association.

SENSEI-ASG: A second dataset for the task of ASGP. We created this dataset due to the fact that our previously developed dataset, Debatabase-ASG, being based around a curated debate website, is somewhat artificial and unrepresentative of more informal debates which occur online and face-to-face, outside of formal settings. Although it contains fewer ASGs when compared to Debatabase-ASG, it is richer in several other respects. It contains more variation in graph structures, particularly much deeper tree structures, as well as more realistic, informal language. These aspects make ASGP on this dataset an extremely challenging task, and therefore, hopefully, this dataset will stimulate further investigation into how to improve ASGP.

10.4 Experimental Results

Reasoning Marker Detection: (Chapter 4) We compare the performance of two language models on this task, BERT and t5, and find that BERT performs better. Additionally, we experiment with adding information about ADU types in the form of special tokens, but found that this lowered performance. The

best performing model was fine-tuned bert-base-uncased, which achieved an F1 score of 0.69.

Argument Structure Parsing (ASP): (Chapter 7) We have replicated the results of Kawarada et al. (2024), showing that end-to-end text-to-text models are a viable option for ASP, as well as showing that a state-of-the-art decoder-only model (LLaMA-3) performs comparably to the encoder-decoder model that was used in their work (Flan-T5). We have additionally shown that an alternative output representation can be used for this task, which achieves comparable results to the representation used by Kawarada et al. (2024), while being significantly more compact and therefore faster to generate.

Argument Summary Graph Parsing (ASGP): Carrying out experiments on our novel task, ASGP, our first major result (Chapter 6) is that a single end-to-end text-to-text model performs better at this task than a pipeline approach consisting of stance detection and summarisation modules.

Secondly, we investigate transferability between our two ASGP datasets, and an ASP dataset (the AAEC). We find that fine-tuning with additional data on top of SENSEI-ASG, either an out-of-genre ASP dataset (the AAEC) or an in-genre ASGP dataset (Debatabase-ASG) can improve results on SENSEI-ASG. However, fine-tuning using *both* additional datasets does not appear to give an added benefit above using one or the other.

10.5 Future Work

At the end of Chapters 4 6, 7, 8 and 9, we have discussed chapter-specific limitations and possible future work to address these. Therefore, here we will

mention only a few broader-scope considerations.

Generation of free-text summaries from Argument Summary Graphs In the introduction, we proposed two separate strands of work that form the basis of this project: the generation of ASGs from dialogues, and the conversion of the generated ASGs into free-text summaries. Owing to the difficulty of both of these tasks, we have mainly addressed the former, and only addressed the latter tangentially in our work on Reasoning Marker Detection in Chapter 4. Therefore, there is much left to investigate around this topic.

To our knowledge, only one previous work has proposed algorithms for the generation of free-text summaries from an argument graph: Barker and Gaizauskas (2016). Due to the progress in NLP since then, there is the clear possibility of developing new techniques for this task. The main obstacles lie in the lack of relevant data resources, and a lack of clarity around how they would be evaluated.

Human Evaluation of Utility of Argument Summary Trees Despite our intuitions about the utility of these tree structures for quickly understanding controversial discourse, it would be useful to carry out some experiments to test these intuitions. Such experiments are inherently difficult to carry out due to the fact that summaries of lengthy comment threads are in themselves quite long, and therefore any experiment which requires human participants to read them is likely to be expensive. Additionally, it is unclear at the moment if there is an objective way for participants to evaluate the utility of a given type of summary.

Bibliography

Rob Abbott, Brian Ecker, Pranav Anand, and Marilyn Walker. Internet Argument Corpus 2.0: an SQL Schema for Dialogic Social Media and the Corpora to Go With It. In *Proc. of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4445–4452, 2016.

Pablo Accuosto and Horacio Saggion. Discourse-driven Argument Mining in Scientific Abstracts. In *International Conference on Applications of Natural Language to Information Systems*, pages 182–194. Springer, 2019.

Stergos Afantenos, Andreas Peldszus, and Manfred Stede. Comparing Decoding Mechanisms for Parsing Argumentative Structures. *Argument & Computation*, 9(3):177–192, 2018.

Meta AI. Introducing Meta Llama 3: the Most Capable Openly Available LLM to Date, 2024. https://ai.meta.com/blog/meta-llama-3/.

Yamen Ajjour, Henning Wachsmuth, Johannes Kiesel, Martin Potthast, Matthias Hagen, and Benno Stein. Data Acquisition for Argument Search: the Args.me Corpus. In *Ki 2019: Advances in Artificial Intelligence: 42nd German Conference on Ai, Kassel, Germany, September 23–26, 2019, Proc. 42*, pages 48–59. Springer, 2019.

Khalid Al-Khatib, Yufang Hou, Henning Wachsmuth, Charles Jochim, Francesca Bonin, and Benno Stein. End-to-end Argumentation Knowledge Graph Construction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7367–7374, 2020.

Khalid Al Khatib, Lukas Trautner, Henning Wachsmuth, Yufang Hou, and Benno Stein. Employing Argumentation Knowledge Graphs for Neural Argument Generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4744–4754, 2021.

Abdullah Al Zubaer, Michael Granitzer, and Jelena Mitrović. Performance Analysis of Large Language Models in the Domain of Legal Argument Mining. *Frontiers in Artificial Intelligence*, 6:1278796, 2023.

Moritz Altemeyer, Steffen Eger, Johannes Daxenberger, Tim Altendorf, Philipp Cimiano, and Benjamin Schiller. Argument Summarization and Its Evaluation in the Era of Large Language Models. *arXiv preprint arXiv:2503.00847*, 2025.

Diego Antognini. Py-rouge. https://pypi.org/project/py-rouge/, 2020. Accessed: 2025-01-28.

Aristotle. *On Rhetoric: a Theory of Civic Discourse*. Oxford University Press, 1991.

J.L. Austin. *How to Do Things With Words*. Oxford University Press, 1962.

Hosein Azarbonyad, Mostafa Dehghani, Tom Kenter, Maarten Marx, Jaap Kamps, and Maarten de Rijke. Hierarchical Re-estimation of Topic Models for Measuring Topical Diversity. In *Advances in Information Retrieval: 39th European Conference on IR Research, ECIR 2017, Aberdeen, Uk, April 8-13, 2017, Proceedings 39*, pages 68–81. Springer, 2017.

Yuntao Bai, Andrew Jones, Kamal Ndousse, et al. Phi-1: Scaling Language Models With ϕ-functions. *arXiv preprint arXiv:2306.11644*, 2023.

Collin F Baker, Charles J Fillmore, and John B Lowe. The Berkeley Framenet Project. In *Coling 1998 Volume 1: the 17th International Conference on Computational Linguistics*, 1998.

Jianzhu Bao, Chuang Fan, Jipeng Wu, Yixue Dang, Jiachen Du, and Ruifeng Xu. A Neural Transition-based Model for Argumentation Mining. In *Proc. of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6354–6364, 2021.

Roy Bar-Haim, Lilach Eden, Roni Friedman, Yoav Kantor, Dan Lahav, and Noam Slonim. From Arguments to Key Points: Towards Automatic Argument Summarization. *arXiv preprint arXiv:2005.01619*, 2020.

Emma Barker and Robert Gaizauskas. Summarizing Multi-party Argumentative Conversations in Reader Comment on News. In *Proceedings of the Third Workshop on Argument Mining (Argmining2016)*, pages 12–20, 2016.

Emma Barker, Monica Lestari Paramita, Ahmet Aker, Emina Kurtić, Mark Hepple, and Robert Gaizauskas. The Sensei Annotated Corpus: Human Summaries of Reader Comment Conversations in On-line News. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 42–52, 2016.

Joe Barrow, Rajiv Jain, Nedim Lipka, Franck Dernoncourt, Vlad Morariu, Varun Manjunatha, Douglas W Oard, Philip Resnik, and Henning Wachsmuth. Syntopical Graphs for Computational Argumentation Tasks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1583–1595, 2021.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: the Long-document Transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Manik Bhandari, Pranav Gour, Atabak Ashfaq, Pengfei Liu, and Graham Neubig. Re-evaluating Evaluation in Text Summarization. *arXiv preprint arXiv:2010.07100*, 2020.

Arne Binder, Tatiana Anikina, Leonhard Hennig, and Simon Ostermann. DFKI-MLST at DialAM-2024 Shared Task: System Description. In *Proceedings of the 11th Workshop on Argument Mining (Argmining 2024)*, pages 93–102, 2024.

Filip Boltužić and Jan Šnajder. Back Up Your Stance: Recognizing Arguments in Online Discussions. In *Proceedings of the First Workshop on Argumentation Mining*, pages 49–58, 2014.

Salvador Pons Bordería. A Functional Approach to the Study of Discourse Markers. In *Approaches to Discourse Particles*, pages 77–99. Brill, 2006.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating Embeddings for Modeling multi-relational Data. *Advances in neural information processing systems*, 26, 2013.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language Models Are Few-shot Learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Yi Bu, Mengyang Li, Weiye Gu, and Win bin Huang. Topic Diversity: a Discipline Scheme-free Diversity Measurement for Journals. *Journal of the Association for Information Science and Technology*, 72(5):523–539, May 2021. doi: 10.1002/asi.24433.

Katarzyna Budzynska, Mathilde Janier, Juyeon Kang, Chris Reed, Patrick Saint-Dizier, Manfred Stede, and Olena Yaskorska. Towards Argument Mining from Dialogue. In *Computational Models of Argument*, pages 185–196. IOS Press, 2014a.

Katarzyna Budzynska, Mathilde Janier, Chris Reed, Patrick Saint-Dizier, Manfred Stede, and Olena Yaskorska. A Model for Processing Illocutionary

Structures and Argumentation in Debates. In *Lrec 2014: Ninth International Conference on Language Resources and Evaluation*, pages 917–924, 2014b.

Katarzyna Budzynska, Mathilde Janier, Chris Reed, and Patrick Saint-Dizier. Theoretical Foundations for Illocutionary Structure Parsing 1. *Argument & Computation*, 7(1):91–108, 2016.

Andrea Calderaro. Social Media and Politics. In W. Outhwaite and S. Turner, editors, *The SAGE Handbook of Political Sociology*, pages 781–796. SAGE, 2017.

Sirawut Chaixanien, Eugene Choi, Shaden Shaar, and Claire Cardie. Pungene at dialam-2024: Identification of Propositional and Illocutionary Relations. In *Proceedings of the 11th Workshop on Argument Mining (Argmining 2024)*, pages 119–123, 2024.

Tuhin Chakrabarty, Christopher Hidey, Smaranda Muresan, Kathy McKeown, and Alyssa Hwang. Ampersand: Argument Mining for Persuasive Online Discussions. *arXiv preprint arXiv:2004.14677*, 2020.

Danqi Chen and Christopher D Manning. A Fast and Accurate Dependency Parser Using Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (Emnlp)*, pages 740–750, 2014.

Guizhen Chen, Liying Cheng, Luu Anh Tuan, and Lidong Bing. Exploring the Potential of Large Language Models in Computational Argumentation. *arXiv preprint arXiv:2311.09022*, 2023.

Yulong Chen, Yang Liu, Liang Chen, and Yue Zhang. Dialogsum: a Real-life Scenario Dialogue Summarization Dataset. *arXiv preprint arXiv:2105.06762*, 2021.

Liying Cheng, Lidong Bing, Qian Yu, Wei Lu, and Luo Si. APE: Argument Pair Extraction from Peer Review and Rebuttal Via Multi-task Learning. In

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (Emnlp), pages 7000–7011, 2020.

Yoeng-Jin Chu. On the Shortest Arborescence of a Directed Graph. *Scientia Sinica*, 14:1396–1400, 1965.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, et al. Scaling Instruction-finetuned Language Models. *arXiv preprint arXiv:2210.11416*, 2022.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling Instruction-finetuned Language Models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

Jonathan Clayton and Robert Gaizauskas. Predicting the Presence of Reasoning Markers in Argumentative Text. In *Proceedings of the 9th Workshop on Argument Mining*, pages 137–142, 2022.

Jonathan Clayton, Marco Damonte, and Robert Gaizauskas. Parsing Graphical Summaries from Argumentative Dialogues. In *Computational Models of Argument*, pages 37–48. IOS Press, 2024.

Jonathan Clayton, Marco Damonte, and Robert Gaizauskas. SENSEI-ASG: A Challenging Dataset for Argument Summary Graph Parsing. In *Proceedings of the 15th Language Resources and Evaluation Conference (LREC 2026)*, 2026. Accepted for publication.

Hamish Croser. clause-segmenter. https://pypi.org/project/clause-segmenter/, 2023. Python package.

Adrian de Wynter and Tangming Yuan. "i'd Like to Have an Argument, Please": Argumentative Reasoning in Large Language Models. In *Computational Models of Argument*, pages 73–84. IOS Press, 2024.

Liesbeth Degand and Ted Sanders. The Impact of Relational Markers on Expository Text Comprehension in L1 and L2. *Reading and writing*, 15:739–757, 2002.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient Finetuning of Quantized LLMs. *arXiv:2305.14314*, 2023.

Daniel Deutsch and Dan Roth. Understanding the Extent to Which Content Quality Metrics Measure the Information Quality of Summaries. In *Proceedings of the 25th Conference on Computational Natural Language Learning*, pages 300–309, 2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805*, 2018.

Gabriela Andrea Diaz, Carlos Iván Chesñevar, Elsa Estevez, and Ana Maguitman. Stance Trees: a Novel Approach for Assessing Politically Polarized Issues in Twitter. In *Proceedings of the 15th International Conference on Theory and Practice of Electronic Governance*, pages 19–24, 2022.

Deborah Dore, Stefano Faralli, and Serena Villata. Leveraging Graph Structural Knowledge to Improve Argument Relation Prediction in Political Debates. In *Proceedings of the 12th Argument Mining Workshop*, pages 74–86, 2025.

T Dozat. Deep Biaffine Attention for Neural Dependency Parsing. *arXiv preprint arXiv:1611.01734*, 2016.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The Llama-3 Herd of Models. *arXiv preprint arXiv:2407.21783*, 2024.

Phan Minh Dung. On the Acceptability of Arguments and Its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-person Games. *Artificial intelligence*, 77(2):321–357, 1995.

Judith Eckle-Kohler, Roland Kluge, and Iryna Gurevych. On the Role of Discourse Markers for Discriminating Claims and Premises in Argumentative Discourse. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2236–2242, 2015.

Jack Edmonds et al. Optimum Branchings. *Journal of Research of the national Bureau of Standards B*, 71(4):233–240, 1967.

Charlie Egan, Advaith Siddharthan, and Adam Wyner. Summarising the Points Made in Online Political Debates. In *Proceedings of the 3rd Workshop on Argument Mining, the 54th Annual Meeting of the Association for Computational Linguistics*, pages 134–143. Association for Computational Linguistics (ACL), 2016.

Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. Neural End-to-end Learning for Computational Argumentation Mining. *arXiv:1704.06104*, 2017.

James B Freeman. *Dialectics and the Macrostructure of Arguments: a Theory of Argument Structure*, volume 10. Walter de Gruyter, 1991.

James B Freeman. *Argument Structure: Representation and Theory*, volume 18. Springer Science & Business Media, 2011.

Debela Gemechu and Chris Reed. Decompositional Argument Mining: a General Purpose Approach for Argument Graph Construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 516–526, 2019.

Debela Gemechu and Chris Reed. External Knowledge-Driven Argument Mining: Leveraging Attention-Enhanced Multi-Network Models. In *2024*

Conference on Empirical Methods in Natural Language Processing, pages 3688–3709. Association for Computational Linguistics (ACL), 2024.

Debela Gemechu and Chris Reed. CU-MAM: Coherence-Driven Unified Macro-Structures for Argument Mining. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 19731–19749, 2025.

Deniz Gorur, Antonio Rago, and Francesca Toni. Can Large Language Models Perform Relation-based Argument Mining? *arXiv:2402.11243*, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*, 2024.

Ivan Habernal and Iryna Gurevych. Which Argument Is More Convincing? Analyzing and Predicting Convincingness of Web Arguments Using Bidirectional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1589–1599, 2016.

Ivan Habernal and Iryna Gurevych. Argumentation Mining in User-generated Web Discourse. *Computational linguistics*, 43(1):125–179, 2017.

Ivan Habernal, Henning Wachsmuth, Iryna Gurevych, and Benno Stein. The Argument Reasoning Comprehension Task: Identification and Reconstruction of Implicit Warrants. In Marilyn Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1930–1940, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1175. URL https://aclanthology.org/N18-1175/.

Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring Network Structure, Dynamics, and Function Using Networkx. In Gäel Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference (Scipy2008)*, pages 11–15, Pasadena, CA USA, 2008.

Udo Hahn and Inderjeet Mani. The Challenges of Automatic Summarization. *Computer*, 33(11):29–36, 2000.

Annette Hautli-Janisz, Zlata Kikteva, Wassiliki Siskou, Kamila Gorska, Ray Becker, and Chris Reed. QT30: a Corpus of Argument and Conflict in Broadcast Debate. In *Proceedings of the 13th Language Resources and Evaluation Conference*, pages 3291–3300. European Language Resources Association (ELRA), 2022.

Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, et al. Knowledge Graphs. *ACM Computing Surveys (Csur)*, 54(4):1–37, 2021.

Matthew Honnibal, Ines Montani, et al. spaCy: Industrial-strength Natural Language Processing in Python. https://spacy.io, 2023.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank Adaptation of Large Language Models. *arXiv preprint arXiv:2106.09685*, 2021.

Jingyuan Huang, Quntian Fang, Sijie Wang, Zhiliang Tian, Feng Liu, Zhen Huang, and Dongsheng Li. Argumentative Relationship Recognition Based on End-to-end Multitask Learning, 2023. Tentative proceedings of the 12th International Joint Conference on Knowledge Graphs.

Arman Irani, Ju Yeon Park, Kevin Esterling, and Michalis Faloutsos. WIBA: What Is Being Argued? a Comprehensive Approach to Argument Mining. *arXiv preprint arXiv:2405.00828*, 2024.

Terne Sasha Thorn Jakobsen, Maria Barrett, and Anders Søgaard. Spurious Correlations in Cross-topic Argument Mining. In *Proceedings of* Sem 2021: the Tenth Joint Conference on Lexical and Computational Semantics*, pages 263–277, 2021.

Mojan Javaheripi, Sébastien Bubeck, Marah Abdin, Jyoti Aneja, Sebastien Bubeck, Caio César Teodoro Mendes, Weizhu Chen, Allie Del Giorno, Ronen Eldan, Sivakanth Gopi, et al. Phi-2: the Surprising Power of Small Language Models. *Microsoft Research Blog*, 2023.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. Draft version, available online at https://web.stanford.edu/ jurafsky/slp3/, 3rd edition, 2023. Accessed: 2025-07-10.

Masayuki Kawarada, Tsutomu Hirao, Wataru Uchida, and Masaaki Nagata. Argument Mining As a Text-to-text Generation Task. In *Eacl Proceedings (Volume 1: Long Papers)*, pages 2002–2014, 2024.

Mohammad Khosravani, Chenyang Huang, and Amine Trabelsi. Enhancing Argument Summarization: Prioritizing Exhaustiveness in Key Point Generation and Introducing an Automatic Coverage Evaluation Metric. *arXiv preprint arXiv:2404.11793*, 2024.

Kialo. Kialo.com, 2024. URL http://www.kialo.com.

Zlata Kikteva, Alexander Trautsch, Patrick Katzer, Mirko Oest, Steffen Herbold, and Annette Hautli. On the Impact of Reconstruction and Context for Argument Prediction in Natural Debate. In *Proceedings of the 10th Workshop on Argument Mining*, pages 100–106, 2023.

Diederik P Kingma and Jimmy Ba. Adam: a Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alistair Knott. A Data-driven Methodology for Motivating a Set of Coherence Relations. 1996.

Harold W. Kuhn. The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97, 1955.

Tatsuki Kuribayashi, Hiroki Ouchi, Naoya Inoue, Paul Reisert, Toshinori Miyoshi, Jun Suzuki, and Kentaro Inui. An Empirical Study of Span Representations in Argumentation Structure Parsing. In *Acl Proceedings*, pages 4691–4698, 2019.

John Lawrence and Chris Reed. Combining Argument Mining Techniques. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 127–136, 2015.

John Lawrence and Chris Reed. Argument Mining: a Survey. *Computational Linguistics*, 45(4):765–818, 2020.

Vladimir I Levenshtein et al. Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. In *Soviet Physics Doklady*, volume 10, pages 707–710, 1966.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing Continuous Prompts for Generation. *arXiv preprint arXiv:2101.00190*, 2021.

Chin-Yew Lin. Rouge: a Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, pages 74–81, 2004.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, Prompt, and Predict: a Systematic Survey of Prompting Methods in Natural Language Processing. *arXiv preprint*, 2021. URL https://arxiv.org/abs/2107.13586.

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. *arXiv preprint arXiv:1711.05101*, 2017. URL https://arxiv.org/abs/1711.05101.

Eric Malmi, Daniele Pighin, Sebastian Krause, and Mikhail Kozhevnikov. Automatic Prediction of Discourse Connectives. *arXiv preprint arXiv:1702.00992*, 2017.

William C Mann and Sandra A Thompson. Rhetorical Structure Theory: Toward a Functional Theory of Text Organization. *Text-interdisciplinary Journal for the Study of Discourse*, 8(3):243–281, 1988.

Tobias Mayer, Santiago Marro, Elena Cabrio, and Serena Villata. Generating Adversarial Examples for Topic-dependent Argument Classification 1. In *Computational Models of Argument*, pages 33–44. IOS Press, 2020.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. Non-projective Dependency Parsing Using Spanning Tree Algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, 2005.

Leland McInnes, John Healy, Steve Astels, et al. Hdbscan: Hierarchical Density Based Clustering. *J. Open Source Softw.*, 2(11):205, 2017.

Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, 2013.

Amita Misra, Pranav Anand, Jean E Fox Tree, and Marilyn Walker. Using Summarization to Discover Argument Facets in Online Ideological Dialog. *arXiv preprint arXiv:1709.00662*, 2017.

Gaku Morio, Hiroaki Ozaki, Terufumi Morishita, Yuta Koreeda, and Kohsuke Yanai. Towards Better Non-tree Argument Mining: Proposition-level Biaffine Parsing With Task-specific Parameterization. In *Acl Proceedings*, pages 3259–3266, 2020.

Gaku Morio, Hiroaki Ozaki, Terufumi Morishita, and Kohsuke Yanai. End-to-end Argument Mining With Cross-corpora Multi-task Learning. *Transactions of the Association for Computational Linguistics*, 10:639–658, 2022.

Marius Mosbach, Tiago Pimentel, Shauli Ravfogel, Dietrich Klakow, and Yanai Elazar. Few-shot fine-tuning Vs. in-context Learning: a Fair Comparison and Evaluation. *arXiv preprint arXiv:2305.16938*, 2023.

Yida Mu, Chun Dong, Kalina Bontcheva, and Xingyi Song. Large Language Models Offer an Alternative to the Traditional Approach of Topic Modelling. *arXiv preprint arXiv:2403.16248*, 2024.

Shikhar Murty, Pratyusha Sharma, Jacob Andreas, and Christopher D Manning. Grokking of Hierarchical Structure in Vanilla Transformers. *arXiv preprint arXiv:2305.18741*, 2023.

Saul B Needleman and Christian D Wunsch. A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins. *Journal of molecular biology*, 48(3):443–453, 1970.

Ani Nenkova, Lucy Vanderwende, and Kathleen McKeown. A Compositional Context Sensitive Multi-document Summarizer: Exploring the Factors That Influence Summarization. In *Proceedings of the 29th Annual International Acm Sigir Conference on Research and Development in Information Retrieval*, pages 573–580, 2006.

Vlad Niculae, Joonsuk Park, and Claire Cardie. Argument Mining With Structured Svms and Rnns. *arXiv preprint arXiv:1704.06869*, 2017.

Timothy Niven and Hung-Yu Kao. Probing Neural Network Comprehension of Natural Language Arguments. *arXiv preprint arXiv:1907.07355*, 2019.

Joakim Nivre. Algorithms for Deterministic Incremental Dependency Parsing. *Computational Linguistics*, 34(4):513–553, 2008.

Sarah Louise Oates. Multiple Discourse Marker Occurrence: Creating Hierarchies for Natural Language Generation. In *Proceedings of the Anlp-naacl 2000 Student Research Workshop*, 2000.

OpenAI. Chatgpt: Optimizing Language Models for Dialogue, 2022. https://openai.com/blog/chatgpt/.

Raquel Mochales Palau and Marie-Francine Moens. Argumentation Mining: the Detection, Classification and Structure of Arguments in Text. In *Proceedings of the 12th International Conference on Artificial Intelligence and Law*, pages 98–107, 2009.

Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, Rishita Anubhai, Cicero Nogueira dos Santos, Bing Xiang, and Stefano Soatto. Structured Prediction As Translation Between Augmented Natural Languages. *arXiv:2101.05779*, 2021.

Joonsuk Park and Claire Cardie. A Corpus of Erulemaking User Comments for Measuring Evaluability of Arguments. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (Lrec 2018)*, 2018.

Gary Patterson and Andrew Kehler. Predicting the Presence of Discourse Connectives. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 914–923, 2013.

Andreas Peldszus and Manfred Stede. From Argument Diagrams to Argumentation Mining in Texts: a Survey. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, 7(1):1–31, 2013.

Andreas Peldszus and Manfred Stede. An Annotated Corpus of Argumentative Microtexts. In *Argumentation and Reasoned Action: Proc. of the 1st European Conference on Argumentation, Lisbon*, volume 2, pages 801–815, 2015a.

Andreas Peldszus and Manfred Stede. Joint Prediction in MST-style Discourse Parsing for Argumentation Mining. In *Proc. of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 938–948, 2015b.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (Emnlp)*, pages 1532–1543, 2014.

Isaac Persing and Vincent Ng. End-to-end Argumentation Mining in Student Essays. In *Naacl: Hlt Proceedings*, pages 1384–1394, 2016.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks. *arXiv preprint*, 2019. URL https://arxiv.org/abs/1903.05987.

Massimo Poesio, Jon Chamberlain, Udo Kruschwitz, Livio Robaldo, and Luca Ducceschi. Phrase Detectives: Utilizing Collective Intelligence for Internet-scale Language Resource Creation. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 3(1):1–44, 2013.

Peter Potash, Alexey Romanov, and Anna Rumshisky. Here's My Point: Joint Pointer Architecture for Argument Mining. *arXiv:1612.08994*, 2016.

Prakash Poudyal, Jaromír Šavelka, Aagje Ieven, Marie Francine Moens, Teresa Gonçalves, and Paulo Quaresma. Echr: Legal Corpus for Argument Mining. In *Proceedings of the 7th Workshop on Argument Mining*, pages 67–75, 2020.

Jan Wira Gotama Putra, Simone Teufel, and Takenobu Tokunaga. Annotating Argumentative Structure in English-as-a-foreign-language Learner Essays. *Natural Language Engineering*, 28(6):797–823, 2022.

Muhammad Qorib, Geonsik Moon, and Hwee Tou Ng. Are Decoder-only Language Models Better Than Encoder-only Language Models in Under-

standing Word Meaning? In *Findings of the Association for Computational Linguistics Acl 2024*, pages 16339–16347, 2024.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the Limits of Transfer Learning With a Unified Text-to-text Transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Model-agnostic Interpretability of Machine Learning. *arXiv preprint arXiv:1606.05386*, 2016.

Andrea Rocci and Costanza Lucchini. Diagramming the Enthymematic Structure of Counterarguments. an Introduction to IAMT Diagrams. *Argumentation et Analyse du Discours*, (34), 2025.

Gil Rocha, Henrique Lopes Cardoso, Jonas Belouadi, and Steffen Eger. Cross-genre Argument Mining: Can Language Models Automatically Fill in Missing Discourse Markers? *Argument & Computation*, (Preprint):1–41, 2023.

Allen Roush and Arvind Balaji. Debatesum: a Large-scale Argument Mining and Summarization Dataset. *arXiv:2011.07251*, 2020.

Allen Roush, Yusuf Shabazz, Arvind Balaji, Peter Zhang, Stefano Mezza, Markus Zhang, Sanjay Basu, Sriram Vishwanath, Mehdi Fatemi, and Ravid Shwartz-Ziv. Opendebateevidence: a Massive-scale Argument Mining and Summarization Dataset. *arXiv preprint arXiv:2406.14657*, 2024.

Ramon Ruiz-Dolz, Jose Alemany, Stella M Heras Barberá, and Ana García-Fornes. Transformer-based Models for Automatic Identification of Argument Relations: a cross-domain Evaluation. *IEEE Intelligent Systems*, 36(6): 62–70, 2021a.

Ramon Ruiz-Dolz, Montserrat Nofre, Mariona Taulé, Stella Heras, and Ana García-Fornes. Vivesdebate: a New Annotated Multilingual Corpus of Argumentation in a Debate Tournament. *Applied Sciences*, 11(15):7160, 2021b.

Ramon Ruiz-Dolz, John Lawrence, Ella Schad, and Chris Reed. Overview of Dialam-2024: Argument Mining in Natural Language Dialogues. In *Proceedings of the 11th Workshop on Argument Mining (Argmining 2024)*, pages 83–92, 2024.

Alberto Sanfeliu and King-Sun Fu. A Distance Measure Between Attributed Relational Graphs for Pattern Recognition. *IEEE trans. on systems, man, and cybernetics*, (3):353–362, 1983.

Ekaterina Saveleva, Volha Petukhova, Marius Mosbach, and Dietrich Klakow. Graph-based Argument Quality Assessment. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pages 1268–1280, 2021.

Deborah Schiffrin. *Discourse Markers*. Number 5. Cambridge University Press, 1987.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling Relational Data With Graph Convolutional Networks. In *The Semantic Web: 15th International Conference, Eswc 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*, pages 593–607. Springer, 2018.

John Searle. *Speech Acts: an Essay in the Philosophy of Language*. Cambridge University Press, 1969.

Christian Stab and Iryna Gurevych. Annotating Argument Components and Relations in Persuasive Essays. In *Proceedings of Coling 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1501–1510, 2014a.

Christian Stab and Iryna Gurevych. Identifying Argumentative Discourse Structures in Persuasive Essays. In *Proc. of the 2014 Conference on Empirical Methods in Natural Language Processing (Emnlp)*, pages 46–56, 2014b.

Christian Stab and Iryna Gurevych. Parsing Argumentation Structures in Persuasive Essays. *Computational Linguistics*, 43(3):619–659, 2017.

Christian Stab, Johannes Daxenberger, Chris Stahlhut, Tristan Miller, Benjamin Schiller, Christopher Tauchmann, Steffen Eger, and Iryna Gurevych. Argumentext: Searching for Arguments in Heterogeneous Sources. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 21–25, 2018a.

Christian Stab, Tristan Miller, and Iryna Gurevych. Cross-topic Argument Mining from Heterogeneous Sources Using Attention-based Neural Networks. *arXiv preprint arXiv:1802.05758*, 2018b.

Manfred Stede, Stergos Afantenos, Andreas Peldzsus, Nicholas Asher, and Jérémy Perret. Parallel Discourse Annotations on a Corpus of Short Texts. In *10th International Conference on Language Resources and Evaluation (Lrec 2016)*, pages 1051–1058, 2016.

Manfred Stede, Jodi Schneider, and Graeme Hirst. *Argumentation Mining*. Springer, 2019.

Robert Endre Tarjan. Finding Optimum Branchings. *Networks*, 7(1):25–35, 1977.

Francesca Toni. Logical Theories and Abstract Argumentation: a Survey of Existing Works. *Argument and Computation*, 11(1-2):41–102, 2020. doi: 10.3233/aac-190476.

Stephen E Toulmin. *The Uses of Argument*. Cambridge university press, 2003.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open Foundation and Fine-tuned Chat Models. *arXiv:2307.09288*, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, et al. Llama 2: Open Foundation and Fine-tuned Chat Models. *arXiv preprint arXiv:2307.09288*, 2023b.

Lifu Tu, Garima Lalwani, Spandana Gella, and He He. An Empirical Study on Robustness to Spurious Correlations Using Pre-trained Language Models. *Transactions of the Association for Computational Linguistics*, 8:621–633, 2020.

Matej Ulčar and Marko Robnik-Šikonja. Sequence-to-sequence pretraining for a less-resourced slovenian language. *Frontiers in Artificial Intelligence*, 6: 932519, 2023.

C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 2nd edition, 1979. URL http://www.dcs.gla.ac.uk/Keith/Preface.html. F-measure defined on p. 114.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *Advances in neural information processing systems*, 30, 2017.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer Networks. *arXiv preprint arXiv:1506.03134*, 2015.

Jacky Visser, John Lawrence, Jean Wagemans, and Chris Reed. An Annotated Corpus of Argument Schemes in Us Election Debates. In *Proceedings of the 9th Conference of the International Society for the Study of Argumentation (ISSA), 3-6 July 2018*, pages 1101–1111, 2019.

Jacky Visser, John Lawrence, Chris Reed, Jean Wagemans, and Douglas Walton. Annotating Argument Schemes. *Argumentation*, 35(1):101–139, 2021.

Luis Von Ahn. Games With a Purpose. *Computer*, 39(6):92–94, 2006.

DN Walton. *Argumentation Schemes*. Cambridge University Press, 2008.

Ashley Williams. Using Reasoning Markers to Select the More Rigorous Software Practitioners' Online Content When Searching for Grey Literature.

In *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018*, pages 46–56, 2018.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's Transformers: State-of-the-art Natural Language Processing. *arXiv preprint arXiv:1910.03771*, 2019.

Yuetong Wu, Yukai Zhou, Baixuan Xu, Weiqi Wang, and Yangqiu 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)*, pages 103–109, 2024.

Yuxiao Ye and Simone Teufel. End-to-end Argument Mining As Biaffine Dependency Parsing. In *Proc. of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 669–678, 2021.

Yuxiao Ye and Simone Teufel. Computational Modelling of Undercuts in Real-world Arguments. In *Proceedings of the 11th Workshop on Argument Mining (Argmining 2024)*, pages 59–68, 2024.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating Text Generation With BERT. *arXiv preprint arXiv:1904.09675*, 2019.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M Meyer, and Steffen Eger. Moverscore: Text Generation Evaluating With Contextualized Embeddings and Earth Mover Distance. *arXiv preprint arXiv:1909.02622*, 2019.

George Kingsley Zipf. Human Behavior and the Principle of Least Effort: an Introduction to Human Ecology. 1949.

Appendix A

Technical Details

A.1 Calculation of Topic Diversity Index

Our methodology is as follows: we first segment each text, into paragraphs in the case of the article, and into comments in the case of the comments section.

We then use the OpenAI API to prompt *gpt-3.5-turbo-instruct* with the following prompt, for each segment: `f"What is the topic of the following text? Answer in strictly ONE WORD.\n\n{text}"`

Topical diversity is then calculated as follows:

$$\text{topical\_diversity} = \frac{\text{n\_unique\_topics}}{\text{total\_n\_topics}} \tag{A.1}$$

Since the choice of comments/ paragraphs is an arbitrary segmentation, we experimented with different units, including sentences and clauses. For sentence tokenisation, we used the `SpaCy` Python package's `en_core_web_sm` module (Honnibal et al., 2023). For clause segmentation, we used `clause-segmenter` (Croser, 2023). For both of these segmentation types, there was significantly more diversity in the comments instead of the articles (Welch's t-test: sentences, $p < 0.001$, Cohen's d=1.54; clauses, $p < 0.001$, Cohen's d= 1.63).

A.2 Model Parameters for Chapter 6

Parameter	Longformer (LED)	Phi-2	Llama-2
Base model	allenai/led-large-16384	microsoft/phi-2	meta-llama/Llama-2-7B
Optimizer	AdamW	AdamW	AdamW
Batch size	2	2	2
Max length	16384	2048	4096
Dropout	0.1	0.1	0.1
AdamW beta1	0.9	0.9	0.9
AdamW beta2	0.999	0.999	0.999

Table A.1: Training parameters for Longformer (LED), Phi-2, and Llama-2.

Parameter	Phi-2	Llama-2
r	128	256
lora_alpha	32	64
bias	none	none
task_type	CAUSAL_LM	CAUSAL_LM
target_modules	q_proj, v_proj, k_proj, dense	q_proj, up_proj, o_proj, k_proj, down_proj, gate_proj, v_proj
load_in_4bit	True	True
bnb_4bit_quant_type	nf4	nf4
bnb_4bit_use_double_quant	True	True
bnb_4bit_compute_dtype	torch.bfloat16	torch.float16

Table A.2: QLORA parameters for fine-tuning Phi-2 and Llama-2. (Longformer (LED) does not use QLoRA.)

A.3 Model Parameters for Chapters 7 and 9

Parameter	Llama-3	Flan-T5
Base model	meta-llama/Llama-3-8B	google/Flan-T5-XXL
Optimizer	AdamW	AdamW
Batch size	2	2
Max length	7000	7000
Dropout	0.1	0.1
AdamW beta1	0.9	0.9
AdamW beta2	0.999	0.999

Table A.3: Training Parameters

Parameter	Llama-3	Flan-T5-XXL
r	64	64
lora_alpha	16	16
lora_dropout	0.1	0.1
bias	none	none
task_type	CAUSAL_LM	CAUSAL_LM
target_modules	q_proj, up_proj, o_proj, k_proj, down_proj, gate_proj, v_proj	q, v, k, o
load_in_4bit	True	True
bnb_4bit_quant_type	nf4	nf4
bnb_4bit_use_double_quant	True	True
bnb_4bit_compute_dtype	torch.bfloat16	torch.float16

Table A.4: QLORA parameters for fine-tuning models on Llama-3 and Flan-T5-XXL.

A.4 Calculation of Graph Edit Distance

As described in Chapter 9, Section 9.3.1, we used the Graph Edit Distance between pairs of graphs in the corpus as a measure of intra-corpus variation. Our methodology involved randomly selecting a sample of 100 pairs of graphs from the corpus, calculating the GED between each pair of trees, and then taking the mean average.

As we have already described in Section 5.4.2, formally, the Graph Edit Distance between two graphs g_1 and g_2, $GED(g_1, g_2)$ can be defined as

$$GED(g_1, g_2) = \min_{(e_1,...,e_k) \in \mathcal{P}(g_1,g_2)} \sum_{i=1}^{k} c(e_i) \tag{A.2}$$

where $\mathcal{P}(g_1, g_2)$ denotes the set of edit paths transforming g_1 into g_2, $(e_1, ..., e_k)$ is a sequence of graph edit operations comprising an edit path. Possible edit operations include node insertions, deletions and substitutions, as well as edge insertions, deletions and substitutions.

We use the implementation of GED available in the `Networkx` Python package (Hagberg et al., 2008). Unlike in Section 5.4.2, where we defined a version of GED that we use as a scoring metric, in this version of GED we are only interested in measuring the similarity in tree shapes, and therefore we treat node types as interchangeable and do not need a nodeMatch function. Additionally, in this case, we constrain the GED algorithm to only consider edit paths in which the two root nodes are aligned from the start and not subject to edit operations.

Appendix B

Annotation Guidelines

B.1 General Guidelines

In this task, you will be shown pairs of comments from online news comments sections.

Each pair of comments will be labelled “Comment A" and “Comment B". In each case, Comment A and Comment B are written by two different users and Comment B is responding directly to Comment A.

As well as the comments, you will be provided with the topic of the discussion, as well as a short summary of each comment intended to aid your understanding of what each user is trying to convey.

Figure B.1 is an example of the prompt that you will be shown, which shows the discussion topic and two comments, along with a summary of each of those comments. You are prompted with three options to select from to describe the stance of Comment B towards Comment A: Attack, Support or NA.

In general, we encourage you to bias your selection towards “Attack" or “Support" and select “NA" (meaning “Non-argumentative"/ “Neutral") only as a last resort.

Comment A: Who are "the rich"? To some, a person earning 50k is rich. But if they rent in London and have children they won't feel at all rich. Some CIFers define rich as anyone with more money than them. Comment B: I agree with both of your points. I don't think someone earning 50k is rich, probably below average if living in London.	Comment A: There's something wrong if we have parts of our country that people on £50,000 a year can't afford to live in, I earn a decent salary, so does my partner, we're by no means poor but we couldn't afford to live in London even if we wanted to, it's an incredibly unhealthy situation Comment B: Since when, have those earning double the average wage, suddenly become entitled to the consider themselves 'not rich' or what they can't say 'poor' (because it would be unseemly) . What utter bollocks!
Label: **Support**	Label: **Attack**

Table B.1: Uncontroversial examples of support (left) and attack (right) relations between pairs of comments.

- Select "Attack" if the stance of Comment B generally disagrees with the stance of Comment A.
- Select "Support" if the stance of Comment B generally agrees with the stance of Comment A.
- Select "NA" if the comment:
    1. is not argumentative: for example, it is an explanation, a joke, or an insult
    2. otherwise does not give an opinion about the comment it is responding to
    3. is incomprehensible in context (you cannot understand how Comment B is supposed to relate to Comment A).

Discussion Topic: NHS tax is floated by Liberal Democrats to fill £30bn hole

Comment A Summary: Rich - should pay more tax. Low paid workers constantly fear losing home/ assets etc.

Comment A: Great, let the proles squabble amongst themselves which of their number should consider themselves 'rich' (clue anyone who is 'a couple of paychecks away from the street' as they would say in the US, ain't rich), whilst the oligarchs get on with the job of chiselling Olympic sized swimming pools and full-sized ballrooms underneath their London pads and furnishing their yachts with musical waterfalls and flocks of scented sheep and the like. All hail the 'wealth creators' and God forbid that they should pay a fair share of tax, that way lies disaster.

Comment B Summary: Businesses - should pay triple NI tax - people should pay lower income tax

Comment B: Triple the NI for businesses, and lower the income tax. The NI is one tax multinationals can't avoid paying.

Comment B is a reply to Comment A. What is the stance of Comment B towards Comment A? [Attack/ Support/ NA]

Figure B.1: An example of the type of comment pair you will be asked to annotate.

B.2 Further Guidelines

This section contains more specific advice on types of comment pairs which may be more challenging to label. We are interested in the broadest possible definition of argument, so if in doubt it is preferable to label a comment as either "Attack" or "Support" rather than "NA".

Rhetorical Questions: Rhetorical questions (in cases where it is clear that a user is trying to make an argument), count as argumentative. For example, the comment in Table B.2, containing a rhetorical question, counts as an Attack:

Example Comments	**Comment A:** Britain, super killing machine for the rich since 1707, now alienating ourselves further from a global world by having out fingers in all the dirty pies. **Comment B:** By "killing machine" you mean stuff like the Royal Navy outlawing slavery and piracy on the high seas troughout the C19th?	Inflation adjusted, yes, but in terms of affordability of basic commodities, housing etc, how does that work out?
Label	**Attack**	**NA**
Reason For Label	We can tell from the context that this is a rhetorical question/ challenge, and not a request for clarification, hence we count it as argumentative (an attack)	Questions asking for clarification on some point are not the same as rhetorical questions, and don't count as argumentative.

Table B.2: Rhetorical Questions

Critical comments/ insults: Comments containing insults or criticisms of another user should only be considered "argumentative" if they act as an argument in the context of the discussion. For example, calling another user "illiterate" could be an argumentative comment if they are attempting to call into question that user's claim to be an authority on climate science, for example. However, if the insult has no obvious argumentative function

but seems to be nothing more than an example of online harassment, then it should be labeled as neutral. Note that this is not the same as requiring that the comment be “civil" - we acknowledge that non-civil comments may still have an argumentative function.

Example	Again you must work in the public sector if you think you get big redundancy payments and anything more than a state pension. So that means JSA if your unemployed before 65. You need to get out of your cosy world and look beyond your parents that you envy so much.	You sound lovely
Label	**Attack**	**NA**
Reason for Label	This comment contains criticism/ insult but also arguments, therefore we mark it as argumentative	A comment containing just a personal attack, and nothing more, should be marked as NA

Table B.3: Personal Criticism/ Insults

Comments with Mixed Attack/Support relations Where there are parts of Comment B which disagree with Comment A and parts which agree with it, pick the option which seems to express the overall sentiment of the comment. If this is unclear or the user gives equal weight to attacking and supporting different parts of the previous comment, then select “NA".

Example	**Comment A:** He said it was an example of a big nation demonstrating what they do ..... spend countless billions on a vessel that will at best have no aircraft for at least 6-10 years and when there is enough support vessels to defend this hulking lump. Lets gloss over the anti ship ballistic missiles that could render them sitting ducks. **Comment B:** Agree regarding the time scale for fixed wing aircraft, however I'm not so sure about your statement with regards to anti ship missiles. The carrier and the Type 45 destroyer both have SAMPSON radar, the best in the world. Type 45 also has Sea Viper missiles and Phalanx CIWS. Carriers aren't obsolete yet.	**Comment A:** BAe and HMG have been entirely clear on this, that in the event of a Yes vote that is what the outcome will definitely be. The SNP position that the UK would continue to build them in Rosyth is nonsense as a large government investment is needed in the yard for the T26 programme. **Comment B:** I don't dispute your statement, rUK tax payer will just have to pick up the bill if or as and when they do. However According to the unions, Babcock had noted the Scottish government's plan to remodel Rosyth and Faslane after independence, but said it remained unclear whether workloads would be smaller or greater than now.
Label	**Attack**	**NA**
Reason for Label	In this comment pair, although Comment B concedes one of Comment A's points, it is clear that their overall stance on the main topic under discussion (the obsolescence of aircraft carriers) differs from Comment A, so we count this as an attack	In this pair of comments, the stance of Comment B towards Comment A is not clear (there appear to be both attacking and supporting statements), so we mark it as NA.

Table B.4: Comments with Mixed Attack/ Support Relations

Concession-Counterarguments In some cases, an interlocutor may concede the merit of an opponent's argument, but then provide an alternative argument for their side of the debate. Even though they explicitly or implicitly contain a concession, we will still regard them as "attacks" because they attack the primary argument given in the previous comment.

Example	Good point. Against less high tech adversaries, carriers still have merit.
Label	**Attack**
Reason For Label	An argument which concedes on one aspect but nevertheless attacks the main point put forward by the other side

Table B.5: Example of the Concession-Counterargument Pattern

Explanation vs Argument Sometimes it may be ambiguous as to whether a comment is "explaining facts" or making an argument. This should be decided by the context of the explanation within the comment thread; in some cases, an explanatory comment is posted merely because the comment thought that the facts that it contained could be of interest to other users, whereas in other cases, an explanation is posted in order to try and persuade another user to accept his or her perspective.

Example	I am Welsh, I live in Wales. I worry greatly about the version of History taught to us. We do not teach ourselves enough of the ills, violence and grievous nature of Empire, we have plowed on with expansionist economic policy which itself morphed out of empire. [LONG EXPLANATION WITH EXAMPLES] It will not bode well for us in a global world that is changing rapidly, either to continue our pursuit of aggressive foreign policy or to ignore the ills of the past, we must as a culture evolve peacefully or face destruction - that is my view.	**Comment A:** Saying that the floods were hailed as a "once-in-100-years" event, is misleading. 1:100 year flood is a technical way of describing the scale of the flood, and doesn't mean that a similar size event could not occur 2 years later. **Comment B:** The input timescale is almost certainly not 100 years of data, so that the probability is a pure guess; more likely its an extrapolation based on the typical variation in rainfall observed over say 50 years. Given the Earth is 2 billion years old the nature of these assumptions is easy to black-swan-the-fuck-outta-here. Still, its the most useful and workable measure we have.
Label	**Attack**	**NA**
Reason for Label	This is a comment which contains an element of explanation, but since this explanation is ultimately used in service of putting forward and argument, we still count it as argumentative	In the context, Comment B does not appear to be argumentative, but rather an explanation/ clarification of the point made in Comment A, and therefore we mark it as non-argumentative.

Table B.6: Explanation vs. Argument

Sarcasm Due to the inherently ambiguous nature of sarcasm and the lack of tonal cues in written text, comments which you suspect may contain sarcasm can be difficult to annotate. Due to this it is advisable (unless it is made very clear by the context) to label possibly-sarcastic comments as non-argumentative.

Example	Great, let the proles squabble amongst themselves which of their number should consider themselves 'rich' (clue anyone who is 'a couple of paychecks away from the street' as they would say in the US, ain't rich), whilst the oligarchs get on with the job	Yep... another 18 years without accelerated warming should prove it for all time. And the ocean surface temps... a waste of time and printers ink
Label	**Attack**	**NA**
Reason for Label	The word "great" is obvious sarcasm, made clear by the rest of the comment	It is not clear whether or not this comment is a serious comment denying global warming, or a sarcastic comment mocking climate change denialists, so we mark it a NA.

Table B.7: Sarcastic comments