PolArg: Unsupervised Polarity Prediction of Arguments in Real-Time Online Conversations

Mirko Lenz[1][0000−0002−7720−0436] and Ralph Bergmann[1,2][0000−0002−5515−7158]

[1] Trier University, Universitätsring 15, 54296 Trier, Germany
info@mirko-lenz.de, bergmann@uni-trier.de
[2] German Research Center for Artificial Intelligence (DFKI),
Branch Trier University, Behringstr. 21, 54296 Trier, Germany
ralph.bergmann@dfki.de

Abstract. The increasing usage of social networks has led to a growing number of discussions on the Internet that are a valuable source of argumentation that occurs in real time. Such conversations are often made up of a large number of participants and are characterized by a fast pace. Platforms like X/Twitter and Hacker News (HN) allow users to respond to other users' posts, leading to a tree-like structure. Previous work focused on training supervised models on datasets obtained from debate portals like Kialo where authors provide polarity labels (i.e., support/attack) together with their posts. Such classifiers may yield suboptimal predictions for the noisier posts from X or HN, so we propose unsupervised prompting strategies for large language models instead. Our experimental evaluation found this approach to be more effective for X conversations than a model fine-tuned on Kialo debates, but less effective for HN posts (which are more technical and less argumentative). Finally, we provide an open-source application for converting discussions on these platforms into argument graphs.

Keywords: Argumentation · Argument Graphs · Argument Mining · Large Language Models · Social Networks · Datasets · Open Source

1 Introduction

Argumentation is a fundamental part of human communication and can be found in many different forms. Having the best argument in a conversation is often a key factor to success. Computational Argumentation (CA) consequently has the potential of supporting a wide range of user types—ranging from journalists researching for their next article to students writing their thesis. Most arguments are expressed in natural language, which means that machines first need to parse the argumentative structures within a text through a process called Argument Mining (AM) [18]. With the advent of the Internet, there is a growing number of discussions happening on platforms such as X/Twitter, Reddit, and Hacker News (HN). These new forms of discourse are characterized by a large number of participants and a fast pace and share one common trait: Users can respond

directly to other users' posts, leading to a tree-like structure of the conversation. Compared to plain texts, this allows users to focus on certain parts of the discussion more easily—for instance, by hiding certain parts of the tree.

Discussions on social networks often involve argumentation (e.g., if users try to convince others of their opinion) [14], thus we argue that these platforms are a valuable resource for CA. Imagine an emerging event, such as the release of a new product or a political scandal. In such a situation, it is important to be able to quickly identify the most important arguments—both for experts like journalists and laymen. Curated argumentation databases cannot be used for evolving topics, so this is mostly a manual process at the moment. With the pre-structured conversations from social networks, only two tasks are left to use them as argument graphs: (i) Identifying which of the posts are actually argumentative and (ii) determining the polarity (support or attack) between them. Both have already been tackled in previous work (see Section 3), but existing approaches rely only on supervised classifiers. This means that they need a large amount of *annotated* data to train on, which is not available for social networks like X. Instead, most datasets are obtained from moderated debate portals like Kialo. Contrary to most social networks, posts on these platforms are moderated and users tend to write elaborate replies. The polarity of the replies is explicitly stated, making it relatively easy to train supervised models. We found that the resulting models are not directly applicable to other types of data (e.g., social network posts), requiring the creation of training data from scratch.

To remove the need to annotate social network posts, we propose an unsupervised approach based on prompting strategies for Large Language Models (LLMs). In our paper, we focus on the polarity prediction task, leaving the identification of argumentative posts to future work (see Section 7). Consequently, we pursue the following research question: *Can unsupervised LLMs match or even surpass the polarity prediction quality of supervised approaches?* Our main contributions are (i) Four different prompting strategies for different types of LLMs to predict the polarity between two posts in a conversation, (ii) an extensive experimental evaluation on an existing benchmark corpus as well as two new datasets obtained from the platforms X and HN, and (iii) an open-source application that allows users to perform real-time AM on these two platforms.

The remainder of this paper is structured as follows: Section 2 introduces fundamental concepts, followed by a review of related work in Section 3. Section 4 presents the prompting strategies that are evaluated in Section 5. We discuss limitations in Section 6 and conclude our paper in Section 7.

2 Foundations

In the following section, we introduce the most important concepts of CA and Natural Language Processing (NLP) [2] as well as the conversation platforms used in this paper.

2.1 Computational Argumentation

Before dealing with CA, we start with the concept of an *argument*, often defined as a single *claim* and several supporting or attacking *premises* [21,24]. Both claims and premises are the fundamental elements of argumentation, also known as Argumentative Discourse Units (ADUs) [21], and can range from a few words to complete paragraphs. Most argumentative texts revolve around a primary claim that the author aims to establish, known as the *major claim* [25].

A graph-based format is an intuitive way to represent these structures, leading to the concept of *argument graphs*. In our paper, we use an extended version of the Argument Interchange Format (AIF) [9] and consider it as a triple $G = (V, E, M)$, where all ADUs are nodes or vertices V, the relationships between them form the edges $E \subseteq V \times V$, and M representing its major claim. The graph includes *atom nodes* A representing individual ADUs and *scheme nodes* S denoting the type of connection (i.e., support/attack) between other nodes. Thus, the set of nodes V can be expressed as $V = A \cup S$. In this structure, edges are not allowed to connect two atom nodes by definition, so the set of edges E can be defined as $E = V \times V \setminus A \times A$.

The term AM refers to the process of extracting and identifying argumentative structures from textual data—for instance, detection claims and premises and predicting relations between them. Our work contributes to the latter task, which is also known as *polarity prediction*: "Does a premise *support* or *attack* the claim?" Cayrol and Lagasquie-Schiex [7] introduced a *Bipolar Argumentation Framework* to represent these relations. We stick to the AIF standard and its scheme nodes introduced earlier, so we refer the interested reader to their work for a more formal definition of this framework. By combining multiple AM tasks, complex argument graphs can be constructed.

2.2 Natural Language Processing

The field of NLP offers a wide range of techniques to process natural language texts. When dealing with structured argumentation in the form of graphs, the aforementioned atom nodes contain texts that can be processed through NLP. Since the inception of representing words through embeddings, the concept has evolved to transformer-based models popularized by Bidirectional Encoder Representations from Transformers (BERT) [11]. These models are pre-trained on a large corpus of texts and can then be fine-tuned on a specific task—for instance, predicting Textual Entailment (TE) [16]. TE—also known as Natural Language Inference (NLI)—is the task of determining whether a given text *entails* another text and is conceptually similar to the investigated polarity prediction task. However, datasets for TE are not directly applicable to polarity prediction, since the notion of *entailment/contradiction* is not the same as *support/attack*: For example, a premise may *entail* a claim, but does not necessarily *support* it.

Based on the transformer architecture, LLMs having billions of parameters have been developed in recent years. In addition to fine-tuning, they can be used in a chat-based way by *prompting* them for some output. This approach is also

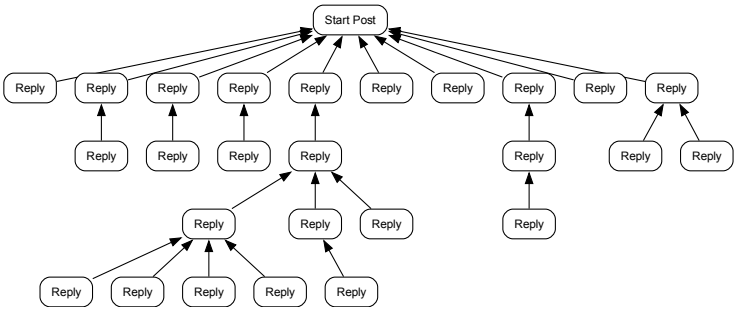

Fig. 1. Fragment of a conversation from the platform Hacker News with the text of the posts replaced by their type.[4]

known as *few-shot learning* [27] since the model does not need to be trained on a large dataset. LLMs differ w.r.t. their maximum context length—that is, the number of tokens they can process at once. As we will discuss in Section 4, this is an important factor to consider when designing prompts.

2.3 Online Conversation Platforms

Having introduced argumentation, the use of graphs in this context, and the most important concepts of LLMs, we now detail the unique characteristics of the different conversation platforms with which we are concerned in this paper: Kialo, X, and HN. The first is a moderated debate portal, whereas the other two are social networks. One common feature is that users can respond to the posts of other users, leading to a tree-like structure of the conversation like the one shown in Figure 1. These trees depict a special type of argument graph, where each scheme node has exactly one incoming and one outgoing connection to some atom node. The starting post of the conversation can be seen as the major claim of the argument graph. Therefore, we can specialize the definition of an argument graph $G = (V, E, M)$ by redefining the set of edges $E = A \times S \cup S \times A$ and setting constraints for the number of outgoing and incoming edges for scheme nodes $\forall s \in S : \text{outdegree}(s) = \text{indegree}(s) = 1$.

A central difference between the three platforms is the type of posts they contain: Kialo[5] is a platform that aims to facilitate high-quality debates by providing a structured environment for users to discuss a wide range of topics. Users not only reply to another user's post, but also explicitly state the polarity of their reply. X[6] (formerly known as *Twitter*) is a social network where users can post short messages (formerly known as *tweets*) that are limited to 280 characters. Similar to Kialo, other users can reply to these tweets, but the polarity or even the stance of their post is unknown. At the same time, X has additional features

[4] Source: `https://news.ycombinator.com/item?id=37744339`

[5] `https://www.kialo.com/`

[6] `https://x.com` and `https://twitter.com`

like *quotes*, *mentions*, and *hashtags* that can be used to refer to other posts. For example, a reply on X may contain multiple mentions of other users, leading to a more complex structure than the hierarchical conversations found on Kialo and HN. In our paper, we focus on the tree-like reply structure and leave the remaining features for future work. HN[7] is a social news website run by the venture capital firm *Y-Combinator* where users can submit links to articles or ask questions and other users can comment on them. The platform is primarily aimed at developers, and discussions are often more technical in nature.

3 Related Work

In the following section, we highlight some of the most important contributions to the field of AM and CA concerned with online conversations. This field of research has received a steady stream of contributions in the last decade, of which we selected the works that are most relevant to our paper. For readers more interested in text mining approaches for tweets that have been proposed in that timeframe, we refer to the study conducted by Karami et al. [15].

The baseline model used in Section 5 is based on the work of Agarwal et al. [1]. The core of their contribution is a deep learning architecture dubbed GraphNLI that predicts the polarity between two posts in a threaded conversation. Instead of relying solely on the textual content of two posts, *graph walks* are used to sample contextual information from nearby nodes in the thread and thus generate richer embeddings. The authors evaluated their approach on debates obtained from the Kialo platform and compared it to four baseline approaches—one of them being a Sentence-BERT (S-BERT) [22] based classifier that only receives the two posts without any context. The results showed that GraphNLI outperformed all baselines on the polarity prediction task, although the difference to the S-BERT classifier in terms of accuracy/precision/recall was rather small (approximately 3%). In an ablation study, the ancestor nodes were found to be relevant to the context than the child nodes. With the graph walks being based on probabilities assigned to nodes, the results are not deterministic. Evaluation of GraphNLI on Twitter data is left for future work.

Other datasets containing argumentative conversations have been proposed in the past—for instance, based on the *Debatepedia* website [5,6]. Bosc et al. [3] created the DART dataset that contains tweets (among other topics) related to the Apple Watch release. At that time, it was not possible to fetch the entire conversation tree, so the authors resorted to heuristics to predict pairs of tweets—meaning that the original structure of the conversation is lost. There also exists a large body of work on AM for social media conversations, ranging from the identification of ADUs [13] to the detection of opinions given some tweet [13,20]. The mentioned DART dataset has been used to identify argumentative tweets and predict their polarity [4] and to recognize facts and sources in tweets [12].

[7] `https://news.ycombinator.com/`

Table 1. Matrix with characteristics of our prompting strategies.

	Isolated	**Sequential**	**Contextualized**	**Batched**
Includes context	×	✓	✓	✓
Parallel predictions	✓	×	✓	✓
Usable without JSON schemas	✓	✓	✓	×
Required context length	small	medium	small	large
Number of predictions for n pairs	n	n	n	1

4 Prompting Strategies

As mentioned in Section 2.2, the use of LLMs shifts the focus from feature engineering and model design to the so-called *prompt engineering*. In the following section, we present four different strategies for predicting the polarity between two posts in a conversation. All of them are *zero-shot* approaches—that is, no exemplary cases are given to the model—since we aim at providing a universally applicable solution for different kinds of conversation. Each strategy is tailored for a different kind of LLM, depending on its capabilities. To estimate the required context size of a model, we distinguish between categories *small* (100–500 tokens), *medium* (500–5,000 tokens), and *large* (more than 5,000 tokens). One strategy makes use of JSON-based responses enforcing a given schema through OpenAI's *function calling* feature, rendering it unusable for other models.

The main difference between the strategies is the amount of context they provide to the model. While the first two strategies only use the tree structure to identify premises and their claims—making them applicable to any kind of conversation—the latter two use it to provide additional information to the model: The *isolated* strategy does not use any context at all, while the *sequential* one provides the model with all previous requests and responses. The *contextualized* strategy samples nearby nodes in the conversation tree, and the *batched* one passes the entire conversation as context. They are designed to work equally well for smaller conversations that have only a few posts, as well as for larger ones that potentially contain hundreds of posts. As part of our evaluation in Section 5, a diverse set of graph sizes is used to verify this. A comparison matrix can be found in Table 1 and concrete prompts in Appendix A.

4.1 Isolated Prompting

In this rather intuitive approach, we simply feed two posts into the model without any additional context from the conversation—that is, we assume that they are self-contained. As part of the system message, the model is instructed to predict the polarity between a premise and its claim and respond with "support" or "attack". This approach can be applied to virtually any LLM and is therefore a good starting point for our evaluation. Since all predictions are separate from each other, one can query the model for all of them in parallel—making inference faster for multi-GPU setups. We observed that LLMs may produce more text

than the single word it is supposed to output, which we deal with by performing substring matching. For example, if the model outputs "I support this claim.", we would consider this as a prediction of "support".

4.2 Sequential Prompting

The basic idea is the same as in the previous approach, but we simulate memory by storing all previous predictions for a single post and passing this conversation history to the model. This could make it possible for the model to provide predictions that are consistent with previous decisions, and thus potentially increase the accuracy. The first prediction for a post still does not have any context—the difference only becomes apparent from the second prediction onward. Since the number of messages increases linearly with the number of posts in a conversation, this strategy is not suitable for LLMs with a limited context size. One can remedy this by removing some of the earlier posts and their predictions from the history, but this would also remove the context for the corresponding posts. Compared to isolated prompting, this strategy cannot be run in parallel.

4.3 Contextualized Prompting

This strategy is an extension of the isolated and sequential prompting approaches that aims to solve their limitations. The isolated technique misses any kind of contextual information, potentially leading to wrong predictions. The sequential approach might be prone to subsequent errors: Wrong predictions for the first requests may influence the model's decision for later ones.

To solve these problems, we propose to sample nearby nodes for contextual information in a similar way to GraphNLI. Agarwal et al. [1] proposed the use of *random* graph walks (see Section 3), which means that the results can change between runs. The authors found that providing four nodes as a context yielded the best results, so we propose the following *deterministic* sampling technique: Choose one parent node of the claim, one child node of the premise, and one sibling node of each (if available), resulting in a maximum of four nearby nodes. In case there are multiple candidates, choose the one with the longest text—this should provide the model with the most information available in nearby nodes. A consequence of this sampling is that some nodes in the graph may have limited context—most notably leaf nodes without siblings—even in large discussions.

While in theory this approach could be applied to both the isolated and sequential prompting, we only use it for the former since the latter already includes context, and we did not find any benefit in combining both techniques. Contextualized prompting will naturally need a larger context length than the isolated approach, but the token size does not scale linearly with the number of posts—consequently, it may be used with LLMs having limited context sizes.

4.4 Batched Prompting

All previous approaches fed the argument pairs to the model one by one, but with the development of LLMs having context sizes of more than 100,000 tokens, we

gain the option of passing all pairs in a single request. It would still be possible to perform a single prediction, but that would be inefficient. Instead, this strategy uses another feature that some LLMs (e.g., those created by OpenAI): The ability to handle structured JavaScript Object Notation (JSON) data—also known as *function calling*. This enables us to use a single request to predict the polarity between all pairs in a conversation. We expect this strategy to show the best efficiency since the model can use the entire conversation as a context.

When querying a LLM with such a complex request, there is a chance of hallucinations—for instance, the model might predict a polarity between two posts that are not connected in the conversation or even come up with posts on its own that are not part of the conversation. In an effort to mitigate these, we append a unique identifier to each premise and claim and use only those predictions that match the corresponding identifiers. In case some predictions are missing, we perform a second request for the missing ones only and provide the available predictions as a context.

5 Experimental Evaluation

In the next section, we present the datasets used for our evaluation, followed by the experimental setup. We then proceed with the results of our experiments and discuss their implications. We start by introducing our hypotheses to answer our research question formulated in Section 1: *Can unsupervised LLMs match or even surpass the polarity prediction quality of supervised approaches?*

H1. The prediction quality of supervised models is influenced by the type of posts in the training data (i.e., debate portals vs. social networks).

H2. Adding context information to the prompts improves the prediction quality of the model.

H3. At least one of our prompting strategies matches or exceeds the prediction quality of established supervised approaches.

H1 aims at showcasing the difficulties in transferring models between different types of posts, whereas H3 checks that our prompting strategies are also applicable to high-quality debates. H2 test which of strategies presented in Section 4 performs best on different types of data.

5.1 Experimental Setup

In order to assess our hypotheses, we implemented our approach in Python and made the source code publicly available on GitHub under the permissive MIT license.[8] Our application is implemented through a client-server architecture, which means that other developers can easily integrate it into their own projects.

[8] `https://github.com/recap-utr/polarg`

To demonstrate this, we built another open-source application called xArguebuf that enables real-time AM on X and HN.[9] Throughout this evaluation, we use a set of standard classification metrics, namely *accuracy* A, *precision* P, and *recall* R. Furthermore, we tested the statistical significance of our results using *McNemar's test* [19] (χ^2 distribution, continuity correction, significance level $\alpha = 0.01$). The test is based on disconcordant pairs in a contingency table and allows us to assess the difference in prediction quality between two approaches when using the same data. Its null hypothesis states that the two approaches are equally good at predicting the polarity between posts.

As LLMs for our evaluation, we use the proprietary ChatGPT developed by OpenAI and the open Llama 2 [26] developed by Facebook. The prompting strategies that require small to medium context length were tested on the `gpt-3.5-turbo-1106` model, whereas the batched one requiring a larger context size was tested on the `gpt-4-1106-preview` model (also known as GPT-4 Turbo). The tests involving Llama all use the model with 13 billion parameters fine-tuned on the chat task. Language models tend to provide unpredictable output, so for each prompt-based evaluation, we provide the number of missing predictions (N/A) as a percentage. Due to load-balancing measures implemented by OpenAI, we could not utilize the full context length of their largest model in a deterministic manner—some requests would randomly time out. For the batched strategy, we thus limited one request to 50 claim-premise pairs and performed multiple requests if necessary.

To compare our prompting strategies with established supervised approaches, we used the same baseline model as Agarwal et al. [1]: A cross-encoder based on the S-BERT architecture.[10] Compared to a regular bi-encoder where the two posts are encoded separately, both posts are passed simultaneously to the transformer. Agarwal et al. [1] report results that almost match their GraphNLI model, so we expect this baseline to be a good indicator for the effectiveness of our prompting strategies. We trained multiple variants of this baseline model on different datasets (see next section) to test H1.

5.2 Datasets

In the following section, we present the three datasets used in our evaluation: Two new ones containing conversations from X and HN as well as the dataset used by Agarwal et al. [1] to evaluate their GraphNLI model. One goal of our paper is to facilitate real-time argument mining, so our methods should be applicable to conversations of different sizes and shapes, including small ones containing only a few posts. An overview of the number of posts contained in them is shown in Figure 2, showing that a wide range of conversation sizes is covered. The part of the GraphNLI dataset used in our evaluation is rather large—conversations on average consist of 86 posts, some even having more than 200 posts—whereas the newly annotated X and HN datasets on average contain 15 and 21 posts,

[9] `https://github.com/recap-utr/xarguebuf`

[10] The same pre-trained model (`distilroberta-base`) is used.

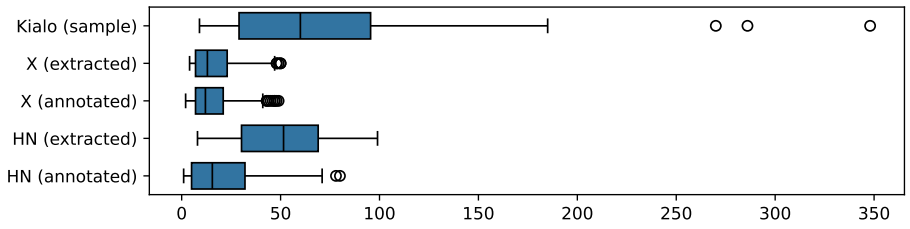

Fig. 2. Distribution of the number of posts in the datasets used in our evaluation.

respectively. Although the X and HN datasets are static snapshots, their diverse sizes and shapes should therefore approximately resemble the conversations that would be encountered in a real-time scenario.

The GraphNLI corpus [1] has been crawled from Kialo and contains a total of 1,560 conversations with 327,579 posts. Since these debates already include polarity labels, manual annotation was not necessary. They also did not need to remove non-argumentative posts from the conversations due given that Kialo is a moderated platform focused on high-quality discourses. Due to the rather large size of the dataset and the rate limits imposed by OpenAI (see above), we sampled 10% of the debates from our test dataset to be used for our evaluation. This test ultimately contains 31 graphs.

The other two datasets containing posts from X and HN have been created from scratch for this paper, as we are not aware of any existing ones that are suitable for our evaluation. After downloading the conversations via the platform's Application Programming Interface (API), the conversation trees were then handed over to two student experts who removed posts that did not contain argumentative content and assigned a polarity (i.e., support/attack) to each missing scheme node. These new corpora are available on request from the authors to other researchers for non-commercial purposes. In the following, we briefly discuss the queries used to obtain the data, the difficulties we faced during the annotation process, and the reliability of the resulting datasets.

To train our baseline classification model, each dataset has been divided into three parts: 80% for training and 20% for testing. The training set was further divided into 80% for training and 20% for validation. The splits were made at the conversation level to ensure that all posts of a single conversation were in the same set to avoid data leakage.

X Dataset This corpus contains posts related to the 2020 presidential election in the United States. Our query is based on hashtags identified in previous studies [23,8]. Here is the complete list of hashtags used in our query:

#2020election, #2020elections, #4moreyears, #americafirst, #biden, #biden2020, #bidenharris2020, #bluewave2020, #covid19, #debate2020, #donaldtrump, #draintheswamp, #election2020, #electionday, #elections_2020, #elections2020, #fourmoreyears, #gop, #joebiden, #kag, #kag2020, #keepamericagreat, #latinosfortrump, #maga, #maga2020, #makeamericagreatagain, #mypresident,

#november3rd, #novemberiscoming, #patriotismwins, #qanon, #redwave, #stopthesteal, #trump, #trump2020, #trump2020landslide, #trumphasnoplan, #trumpliespeopledie, #trumppence2020, #trumpvirus, #uselections, #vote, #vote2020, #votebluetosaveamerica, #votered, #voteredlikeyourlifedependsonit, #voteredtosaveamerica, #votetrump2020, #votetrumpout, #yourchoice, #americafirst

These hashtags were joined together using the logical or operator ($\vee$). Since this query was only used to find the *starting post* of a conversation, we further restricted the set of results using the following constraints: (i) The post must be published between 3rd June 2020 (i.e., start of primaries in Iowa) and 2nd November 2020 (day before election), (ii) is must be written in English, (iii) the author must be verified by Twitter, and (iv) the post must not be a retweet, reply, or quote. When downloading the dataset on 8th December 2022, more than 2,000,000 posts matched these criteria. In other words, we identified over two million conversations, each containing possibly hundreds or even thousands of replies. X's API does not allow filtering by likes, followers, or other metrics, so we decided to let X order the posts by *relevancy* and use the best 500 posts for our annotation process. Our rationale here is that the most relevant posts for X are likely also those that appear in the *For You* tab on their website and app, so this choice should closely mimic the experience of a regular user. For each of the resulting 500 posts, we recursively fetched all replies (i.e., the entire conversation) from X's API, resulting in a file containing more than 2.5 GB of compressed textual data.

Handing over such a large amount of data to our annotators would have been impractical, so we decided to further reduce the number of tweets by applying the following constraints: (i) Each post must have at least 20 characters (otherwise it is unlikely to contain valuable and argumentative information), (ii) each post must have at least one interaction (i.e., like, retweet, reply, or quote), (iii) the depth of a conversation must be at least two (i.e., the distance between the starting post and a leaf reply must be at least two), and (iv) a conversation must afterwards have at least three and at most 50 posts left. The last constraint is necessary to ensure that the annotation process is feasible for our experts. With these restrictions, we were left with 294 conversations that contain 4,930 posts in total. During the annotation process, the experts remove all posts that are not argumentative, leading to a final size of 272 conversations with 4,067 posts. The relatively low number of posts removed during the process again shows that social networks contain a good amount of argumentative content.

Hacker News Dataset We already stated the differences between HN and X in Section 2.3, but it essentially boils down to the fact that HN is a platform targeted at a more technical audience without restrictions on the number of characters. This means that we are not faced with the issue of filtering millions of posts, and thus we used a simpler method to obtain the data. Their API does not natively support arbitrary queries, so we opt to take snapshots of the *best* posts at two different points in time: On 5th October 2023 and 30th October 2023 (about two weeks apart to let enough new posts emerge). We fetched regular

posts and *Ask HN* posts separately and merged them afterwards—there was only one overlapping post between both sets.

But even on HN, this approach resulted in almost 1,000 posts, so we again settled on some constraints to filter out the most promising conversations: (i) The starting post must have received a minimum number of 10 upvotes, (ii) each conversation must have at least ten and at most 100 replies, (iii) each reply must have at least 20 characters, and (iv) the depth of a conversation must be at least two (i.e., the distance between the starting post and a leaf reply must be at least two). These constraints resulted in 206 conversations with 10,596 posts in total. The conversation depicted in Figure 1 is an example of the type of discussion we extracted from the API. After the manual annotation process, we were left with 198 conversations that contained 4,190 posts. This means that more than half of the posts were deemed not argumentative. From our experience, this seems to stem mainly from the fact that the posts on HN contain a lot of factual information instead of opinions. For example, when trying to answer a question in the format *Ask HN*, users are likely to provide a direct answer rather than argue for a certain position. Even if a reply to such a factual post might then contain some argumentative information, we remove the entire branch from the conversation tree to be consistent with the annotation process for the X dataset.

Annotation Reliability During the initial annotation process, each annotator processed a different set of conversations, which means that no Inter-Annotator Agreement (IAA) could be calculated. We also did not have the resources to have each conversation annotated by two experts. To still ensure the reliability of the annotations, we took a random 30% sample of the unannotated X and HN datasets and handed them over to a team of three student experts—more specifically, the team that also labeled the HN dataset. We designed the sampling process in a way that no expert would annotate a conversation they had already seen before. Upon completion, a total of 8,938 scheme nodes had labels by two independent annotators for which we calculated the IAA using Cohen's κ [10].

During the annotation, the experts were free to remove non-argumentative relations, thus we consider two different perspectives: (i) The IAA for the entire dataset (including schemes removed by the annotators), and (ii) the IAA for the subset of scheme nodes that were labeled as either *support* or *attack* by both annotators (i.e., those considered to be argumentative). We received κ values of (i) .434 and (ii) .638 for the X dataset and (i) .202 and (ii) .410 for the HN dataset. Based on the Landis and Koch guidelines [17], we consider the IAA for perspective one (i.e, the entire dataset) to be *moderate* for X and *fair* for HN. For perspective two (i.e., the subset of argumentative schemes), we consider the IAA to be *substantial* for X and *moderate* for HN. The implications of these results are twofold: First, the IAA for the subset of argumentative schemes is higher than for the entire data set, meaning that labeling argumentative content was easier. Second, the IAA for the X dataset is higher than for the HN dataset, indicating X posts are more argumentative HN posts. As stated in Section 1, we leave the detection of argumentative content to future work.

Table 2. Effectiveness results of different variants of the baseline model with the best metrics for each test dataset marked in bold.

Test Dataset	Train Dataset	A	P	R
Kialo	Kialo	.752	.780	.752
Kialo	X ∪ HN	.636	.641	.573
Kialo	All	**.782**	**.785**	**.762**
HN	Kialo	.708	**.642**	.554
HN	X ∪ HN	.700	.612	.612
HN	All	**.715**	.628	**.643**
X	Kialo	.689	.552	.557
X	X ∪ HN	**.753**	**.649**	.625
X	All	.748	.628	**.671**

5.3 Results and Discussion

Having described our experimental setups and the datasets used for our evaluation, we now present our results and discuss them in detail. We start by investigating the effectiveness of our baseline model using Table 2 to answer H1. For the Kialo and HN dataset, the difference between a classifier trained on Kialo graphs and a combination of the three sites is negligible. For posts from X, however, the effectiveness of the model trained on the Kialo dataset is considerably worse than the other two: The model trained on the much smaller X and HN is even the most efficient. Another interesting observation is that this model is considerably less effective on the Kialo test set than the other two. This means that we can only partially confirm H1: Although there is an impact for using Kialo as training data for X posts (and vice versa), HN posts did not show much difference w.r.t. the training data. This seems to strengthen the assumption that the HN posts are more similar to Kialo than they are to X.

The remaining hypotheses can be tested with the results presented in Table 3, starting with the impact of adding context information to the prompts (H2). First, we check whether the contextualized prompting strategy is more effective than the isolated one. For four of our six test cases, we observe a small improvement. However, for the other two, the isolated strategy is more effective. Comparing the isolated and contextualized strategies using McNemar's test yields a p-value of 0.23 across all models and datasets, meaning that the null hypothesis cannot be rejected. Since Agarwal et al. [1] found that adding nearby nodes is beneficial, this could be a consequence of our deterministic sampling method. The results are different for the context-aware batched prompting strategy: For all test cases, we observed notable improvements across all metrics. The comparison of isolated vs. batched and contextualized vs. batched prompting using McNemar's test yields a p-value < 0.001 in both cases, meaning that the null hypothesis can be rejected. This confirms our intuition that passing the whole conversation as context to the model is indeed beneficial. Since this strategy is

Table 3. Effectiveness results of different prompting strategies with LLMs with the best metrics for each test dataset marked in bold.

Test Dataset	Model	Prompting	A	P	R	N/A
Kialo	ChatGPT	Isolated	.593	.564	.720	0.73%
Kialo	ChatGPT	Contextualized	.559	.533	.753	0.15%
Kialo	ChatGPT	Batched	**.840**	**.843**	.841	1.57%
Kialo	Llama	Isolated	.540	.516	**.881**	1.53%
Kialo	Llama	Contextualized	.557	.528	.864	0.04%
HN	ChatGPT	Isolated	.578	.468	.663	0.00%
HN	ChatGPT	Contextualized	.547	.447	.724	0.00%
HN	ChatGPT	Batched	**.618**	**.504**	.686	0.13%
HN	Llama	Isolated	.480	.413	**.827**	0.00%
HN	Llama	Contextualized	.503	.422	.769	0.13%
X	ChatGPT	Isolated	.656	.505	.467	0.34%
X	ChatGPT	Contextualized	.652	.500	.577	0.34%
X	ChatGPT	Batched	**.755**	**.629**	**.752**	2.03%
X	Llama	Isolated	.556	.428	.651	12.77%
X	Llama	Contextualized	.523	.403	.750	4.63%

only possible with the largest GPT model, we cannot compare it to the Llama model. Therefore, we accept H2.

Finally, we check whether our prompting strategies match the effectiveness of the baseline model (H3) by comparing the results of the supervised model trained on all three corpora to the predictions obtained using the batched strategy. For the X dataset, McNemar's test yields a p-value of 0.61 and thus shows that there is no significant difference between the two models. For Kialo and HN, the test yields a p-value < 0.001, leading to the rejection of the null hypothesis. Closer inspection of the classification metrics reveals that in case of Kialo, the batched strategy is more effective than the baseline model, while for HN, the opposite is true. Bearing its low IAA and weak overall scores in mind, this is yet another indicator of the rather technical nature of HN posts—potentially leading to a higher uncertainty in the predictions. Even when considering that ChatGPT may have been trained on some publicly available Kialo debates and may thus be biased towards them, the effectiveness on the new X dataset shows the potential of the batched strategy. Combining all findings, we tend to cautiously confirm H3—at least for clearly argumentative posts.

5.4 Qualitative Error Analysis

Besides the quantitative results, we also performed a qualitative analysis to better understand the errors made by the LLMs. The batched one is the most promising one, so we focus on it in our analysis. Please note that due to copyright issues, we cannot provide examples of actual posts, so we discuss the context and the types of errors made and provide suggestions for improvement.

For the X dataset, we observed that the model often struggles with predictions involving posts that contain insults, sarcasm, or jokes. For example, the polarity between a factual premise and an insulting claim is often predicted differently than by the human annotator. We also identified multiple cases where replies (i.e., the premises) to posts with a negative sentiment (i.e., the claims) were predicted as support by the annotator but as attack by the model. This could be caused by the model comparing the premise to major claim instead of the directly connected claim. Another common source of errors are posts that contain emojis—especially if multiple emojis are used in a single post. Although the experts were able to interpret them correctly, the LLMs may lack the necessary context to do so. One possible solution to this problem could be to encode the emojis via a textual description.

For the HN dataset, we observed the same issues with posts containing insults or negative sentiment. In addition, we found multiple instances where the prediction of the LLMs was different from the expert's opinion, but still plausible or even a better fit. For example, an expert labeled the relation between a premise supporting a claim that in turn attacks another claim as *attack*, while the model correctly predicted *support*. This again shows the inherent subjectivity of the tasks and confirms our finding that the IAA for the HN dataset is lower than for the X dataset (see Section 5.2). For both corpora, we did not observe a correlation between the length of a premise and its claim and the prediction quality of their relation.

One challenge in our analysis was the probabilistic nature of LLMs: Even for the same conversation and prompt, it may happen that the accuracy of the model changes considerably between runs. In order to achieve better stability between the runs, one could modify the prompts to include more specific constraints—potentially at the cost of generalization. This drawback could be mitigated by using specialized prompts for different types of posts.

6 Limitations

While our prompting strategies show promising results, there are still some limitations to consider. Due to rate limits and timeouts imposed by OpenAI, we had to apply chunking to the batched strategy, which may have affected the prediction quality. Additionally, we only consider the text of the posts and do not take into account other modalities like images or videos and thus are missing potentially valuable context information. We also do not analyze links that may be embedded in the posts. In the case of X posts, our current approach focuses on the replies to some starting post, but other relations like mentions or quoted tweets are not considered. Finally, an important aspect to consider is the runtime of the models. While predicting the polarities of a single conversation is a matter of seconds using the supervised model, the LLMs needed almost a minute to complete the task. The reason for this is that such generic LLMs have billions of parameters, while the smaller S-BERT model has millions only. We

expect this to change in the future—even a model like S-BERT was considered to be too slow for use in production just a few years ago.

7 Conclusion & Future Work

In this paper, we presented an unsupervised approach to perform AM on posts from social networks. We introduced multiple prompting strategies for different context lengths and evaluated them on three different datasets. Our results show that the batched prompting strategy—when paired with an adequate LLM—is capable of matching or exceeding the effectiveness of a supervised LLM. Combined with our open-source implementation, this makes it possible to perform real-time AM on social networks even for emerging topics without appropriate training data.

In future work, the presented approach could be extended to also handle the classification of argumentative vs. non-argumentative posts. By adding a neutral class, posts that have little or no argumentative content could be detected and removed from the conversation tree. This could help boost the prediction accuracy, especially for datasets like the HN one where we currently need human annotators to do the job. Another interesting avenue for future work is the evaluation of the LLM Grok developed by xAI. Since this model is specifically trained on posts from X, we expect it to be more effective for this type of data than the generic LLMs used in this paper.

Acknowledgements This work has been funded by the Deutsche Forschungsgemeinschaft (DFG) within the projects *ReCAP* and *ReCAP-II* (№ 375342983, 2018–2024) as part of the priority program *RATIO* (Robust Argumentation Machines, SPP-1999) as well as the *Studienstiftung*.

References

1. Agarwal, V., Young, A.P., Joglekar, S., Sastry, N.: A Graph-Based Context-Aware Model to Understand Online Conversations. ACM Trans. Web (2023)
2. Allen, J.F.: Natural language processing. In: Encyclopedia of Computer Science (2003)
3. Bosc, T., Cabrio, E., Villata, S.: DART: A Dataset of Arguments and their Relations on Twitter. In: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16) (2016)
4. Bosc, T., Cabrio, E., Villata, S.: Tweeties Squabbling: Positive and Negative Results in Applying Argument Mining on Social Media. In: Computational Models of Argument (2016)
5. Cabrio, E., Villata, S.: Detecting Bipolar Semantic Relations among Natural Language Arguments with Textual Entailment: A Study. In: Proceedings of the Joint Symposium on Semantic Processing. Textual Inference and Structures in Corpora (2013)
6. Cabrio, E., Villata, S.: A natural language bipolar argumentation approach to support users in online debate interactions†. Argument & Computation (2013)

7. Cayrol, C., Lagasquie-Schiex, M.C.: On the Acceptability of Arguments in Bipolar Argumentation Frameworks. In: Symbolic and Quantitative Approaches to Reasoning with Uncertainty (2005)
8. Chen, E., Deb, A., Ferrara, E.: #Election2020: The first public Twitter dataset on the 2020 US Presidential election. J Comput Soc Sc (2021)
9. Chesñevar, C.I., McGinnis, J., Modgil, S., Rahwan, I., Reed, C., Simari, G.R., South, M., Vreeswijk, G., Willmott, S.: Towards an argument interchange format. The Knowledge Enigneering Review (2006)
10. Cohen, J.: A Coefficient of Agreement for Nominal Scales. Educational and Psychological Measurement (1960)
11. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs] (2018)
12. Dusmanu, M., Cabrio, E., Villata, S.: Argument Mining on Twitter: Arguments, Facts and Sources. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (2017)
13. Goudas, T., Louizos, C., Petasis, G., Karkaletsis, V.: Argument Extraction from News, Blogs, and Social Media. In: Artificial Intelligence: Methods and Applications (2014)
14. Gurevych, I., Lippi, M., Torroni, P.: Argumentation in Social Media. ACM Trans. Internet Technol. (2017)
15. Karami, A., Lundy, M., Webb, F., Dwivedi, Y.K.: Twitter and Research: A Systematic Literature Review Through Text Mining. IEEE Access (2020)
16. Korman, D.Z., Mack, E., Jett, J., Renear, A.H.: Defining textual entailment. Journal of the Association for Information Science and Technology (2018)
17. Landis, J.R., Koch, G.G.: The Measurement of Observer Agreement for Categorical Data. Biometrics (1977)
18. Lawrence, J., Reed, C.: Argument Mining: A Survey. Computational Linguistics (2019)
19. McNemar, Q.: Note on the sampling error of the difference between correlated proportions or percentages. Psychometrika (1947)
20. Ouertatani, A., Gasmi, G., Latiri, C.: Parsing argued opinion structure in Twitter content. J Intell Inf Syst (2021)
21. Peldszus, A., Stede, M.: From Argument Diagrams to Argumentation Mining in Texts - A Survey. IJCINI (2013)
22. Reimers, N., Gurevych, I.: Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) (2019)
23. Shevtsov, A., Oikonomidou, M., Antonakaki, D., Pratikakis, P., Ioannidis, S.: Analysis of Twitter and YouTube during USelections 2020. arXiv:2010.08183 [cs] (2020)
24. Stab, C., Gurevych, I.: Identifying Argumentative Discourse Structures in Persuasive Essays. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (2014)
25. Stab, C., Gurevych, I.: Parsing Argumentation Structures in Persuasive Essays. Computational Linguistics (2017)
26. Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al.: Llama 2: Open Foundation and Fine-Tuned Chat Models (2023)
27. Wang, S., Fang, H., Khabsa, M., Mao, H., Ma, H.: Entailment as Few-Shot Learner. arXiv:2104.14690 [cs] (2021)

A Prompting Templates

A.1 Isolated Prompting

System You are a helpful assistant that predicts the relation/polarity between the premise and the claim of an argument. You shall predict whether the premise supports or attacks the claim. Answer either `support` or `attack`.
User Premise: `premise`. Claim: `claim`.

A.2 Sequential Prompting

System You are a helpful assistant that predicts the relation/polarity between the premise and the claim of an argument. You shall predict whether the premise supports or attacks the claim. Answer either `support` or `attack`.
User Premise: `premise`. Claim: `claim`.
Assistant `support` *or* `attack`
User Premise: `premise`. Claim: `claim`. And so on...

A.3 Contextualized Prompting

System You are a helpful assistant that predicts the relation/polarity between the premise and the claim of an argument. You shall predict whether the premise supports or attacks the claim. Answer either `support` or `attack`.
User Premise: `premise`. Claim: `claim`. The premise and the claim have the following neighbors in the conversation: `adu_1` ... `adu_n`

A.4 Batched Prompting

System You are a helpful assistant that predicts the relation/polarity between the premise and the claim of an argument. You shall predict whether the premise supports or attacks the claim. Answer either `support` or `attack`. You will be presented with a list of premise-claim pairs containing their text and id encoded as a JSON array. Provide exactly one prediction for each of them using the function `predict_entailment`.
Available Function Calls JSON schema describing `predict_entailment` as an array of objects with the following keys: `premise_id` (string), `claim_id` (string), and `polarity_type` (enum: support/attack).
User JSON array of objects with the following keys: `premise_id` (string), `premise_text` (string), `claim_id` (string), and `claim_text` (string).