NL2FOL: Translating Natural Language to First-Order Logic for Logical Fallacy Detection

Anonymous ACL submission

Abstract

Logical fallacies are common errors in reasoning that undermine the logic of an argument. Automatically detecting logical fallacies has important applications in tracking misinformation and validating claims. In this paper, we design a process to reliably detect logical fallacies by translating natural language to First-order Logic (FOL) step-by-step using Large Language Models (LLMs). We then utilize Satisfiability Modulo Theory (SMT) solvers to reason about the validity of the formula and classify inputs as either a fallacy or valid statement. Our model also provides a novel means of utilizing LLMs to interpret the output of the SMT solver, offering insights into the counter-examples that illustrate why a given sentence is considered a logical fallacy. Our approach is robust, interpretable and does not require training data or fine-tuning. We evaluate our model on a mixed dataset of fallacies and valid sentences. The results demonstrate improved performance compared to end-to-end LLMs, with our classifier achieving an F1-score of 71% on the LOGIC dataset. The approach is able to generalize effectively, achieving an F1-score of 73% on the challenge set, LOGICCLIMATE, outperforming state-of-the-art models by 21% despite its much smaller size.[1]

1 Introduction

A logical fallacy is an argument that may sound convincing, but involves faulty reasoning, leading to an unsupported conclusion (Hamblin, 2022). These fallacies can be committed intentionally to manipulate or spread misinformation, and have been used to spread propaganda in news articles (Musi et al., 2022). Consequently, detecting logical fallacies in natural language text holds a very important potential application in tracking misinformation and validating claims. Recognizing fallacious arguments can make discourse more rational and instructive. In general, logical fallacies could be classified into various types (Jin et al., 2022; van and Francisca, 2017; Tindale, 2007), which are associated with the structure of the sentence. The datasets we use for this research contain 13 different categories of fallacies and examples of some of these are mentioned in Table 1.

As evident, these fallacies evolve out of premises that are not logically sound. They can be identified by a lack of legitimate and relevant evidence that supports their claim. By formally reasoning about these fallacies, we can identify potential issues in the given reasoning effectively. In the last few decades, formal reasoning tools like Boolean satisfiability (SAT) and SMT solvers have advanced considerably. Increases in computing power coupled with algorithmic innovations have enabled major leaps in the capabilities of these solvers, handling millions of variables and functions, heavily complicated logical formulae and numerous theories. Consequently, SMT solvers like Z3 (de Moura and Bjørner, 2008), CVC (Barbosa et al., 2022) have become a key tool in different kinds of program analysis and verification, including studying the satisfiability and validity of logical formulae. These formal reasoning tools allow us to precisely represent arguments symbolically and analyze them to detect logical fallacies through systematic checking for invalid forms of reasoning. This level of rigorous analysis is difficult for humans, so computational tools are useful supplements to scale analysis across large volumes of arguments through methodical application of the rules of deduction and logical calculus.

In order to utilize theory solvers for detecting logical fallacies, it becomes essential to first convert the given statement to logical form. Most of the existing techniques, as discussed in the next section, do not translate natural language sentences to

[1]Our code,data and prompts have been uploaded to the submission system, and will be open-sourced upon acceptance.

Fallacy Name	Example	Logical Form
Faulty Generalization	Sometimes flu vaccines don't work; therefore vaccines are useless.	$(\text{property1}(a) \wedge (a \subseteq b)) \Rightarrow (\forall c \in b\ (\text{property1}(c)))$
False Causality	Every time I wash my car, it rains. Me washing my car has a definite effect on the weather.	$\text{occuredAfter}(a, b) \Rightarrow \text{caused}(a, b)$
Ad Populum	Everyone should like coffee: 95% of teachers do!	$\text{manyPeopleBelieve}(a) \Rightarrow \text{isTrue}(a)$
False Dilemma	I don't want to give up my car, so I don't think I can support fighting climate change.	$\forall(a)(\text{property1}(a) \vee \text{property2}(a))$

Table 1: Few types of logical fallacies along with examples and their logical forms. Note that each type of fallacy may correspond to several logical forms, and the examples provided above are just one possible representation.

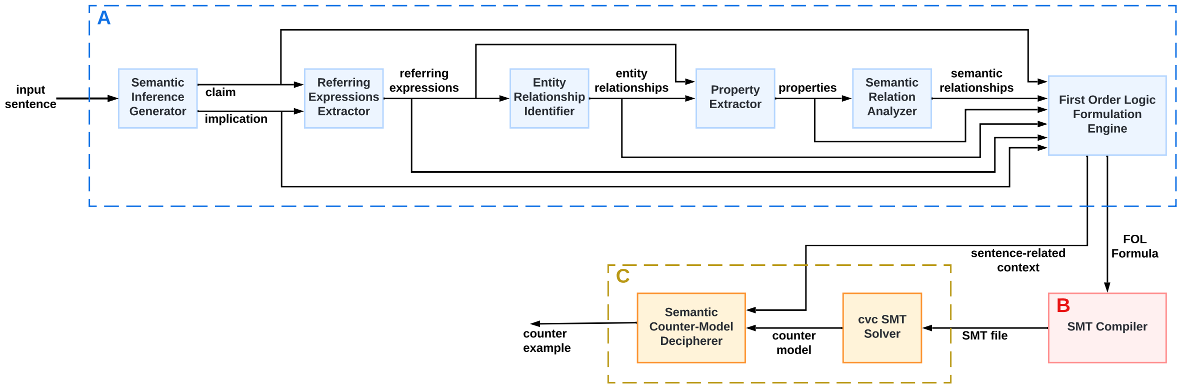

Figure 1: Proposed Logical Fallacy Detection Methodology: *Module A* converts natural language input to a first-order logic formula merged with contextual relationships, *Module B* compiles the negation of a given logical formula to an SMT file with well-defined sorts for variables and predicates, and *Module C* is used to run CVC on the SMT file and if the negation is satisfiable, interpret the counter-model in natural language.

logical form very well. We have developed an effective technique to chain LLMs to translate a given set of statements to first-order logic. Additionally, theory solvers require context, or ground truth, to accurately distinguish logical fallacies from valid statements. This context provides a semantic interpretation of different variables and predicates, without which they have no meaning. Our methodology introduces an effective way to encode that context in a logical formula and utilize it to enrich the theory solver with the necessary context to aid in decision making.

Theory solvers are a good way to identify the validity of a given logical statement. If a set of logical reasoning arguments are invalid, these solvers can be used to obtain a counter-model to the statements. This counter-model serves the explanation behind the faulty reasoning for the statement by providing an interpretation of different variables and predicates where the claims do not lead to the given inference. Counter-models obtained from theory solvers, however, may be hard to interpret because they are in formal notation, which is incomprehensible to a layperson. We have developed an efficient way to utilize LLMs to provide a natural language interpretation of the counter-model, which is more understandable. This helps in further scaling our approach to tracking misinformation in the real world and making it more accessible to everyone.

In this paper, we make the following contributions:

1. We develop an explainable and few-shot method for translating Natural Language to First Order Logic by chaining LLMs

2. We devise a first-order-logic-to-SMT compiler, which, given any string format first-order logic formula, converts it to an SMT file which is fed to the cvc4 solver (Barbosa et al., 2022).

3. We design an effective technique to interpret the results of cvc4 to explain the faulty reasoning behind the sentence in natural language, making it more interpretable.

4. We evaluate our methodology on numerous datasets and prove that it is highly generalizable by testing its effectiveness over a dataset consisting of real-world fallacies related to

climate change.

5. We plan to make our code open-source for the benefit of the research community.

2 Related Work

In this section, we discuss existing research on detection of logical fallacies, converting natural language to first order logic, LLMs and theory solvers.

Logical Fallacy Detection. There have been multiple works on classification of logical fallacies, include classification of argument sufficiency (Stab and Gurevych, 2017), ad hominem fallacies from Reddit posts (Habernal et al., 2018) and dialogues (Sheng et al., 2021), rule parsers (Nakpih and Santini, 2020), structure-aware Transformers (Jin et al., 2022), multitask instruction based prompting (Alhindi et al., 2022) and instance-based reasoning (Sourati et al., 2023). As per our knowledge, our work is the first on classification of logical fallacies in a step-by-step, few shot and explainable manner. This method of making the reasoning process transparent, allowing users to understand and verify the basis on which conclusions are drawn.

Natural Language to Formal Logic Conversion. Early works on natural language to formal logic conversion relied heavily on grammar-based approaches that could handle well-structured language, but struggled with more complex linguistic constructions (Purdy, 1991; Angeli and Manning, 2014; MacCartney and Manning, 2014). These works are hard to generalize because of their inability to work with random sentences without a fixed structure. It has been shown it is hard even for humans to perform such conversions, primarily owing to the ambiguity in natural language (Barker-Plummer et al., 2009).

More recently, advances in neural networks, deep learning and large language models have enabled new data-driven techniques for natural language to linear temporal logic (Cosler et al., 2023; Fuggitti and Chakraborti, 2023; Liu et al., 2022) and first order logic (Singh et al., 2020; Yang et al., 2023; Olausson et al., 2023; Hahn et al., 2022). However, these methods do not provide a way to incorporate ground truth claims, which are necessary for distinguishing logical fallacies from valid sentences. Additionally, owing to the linguistic ambiguity in the English language, most of the approaches have not reached to a level where complex sentences could be accurately transformed to logical form as well as it can be done manually.

Aly et al. (2023) develop an inference pipeline for QA by generating natural logic proofs to identify the relations between claim and evidence text span, in which each proof step is cast into the form of a QA pair. While this work is similar to ours in that it uses a chain of language models to generate a proof and identify relations between two text spans, it requires each proof step to be independent, whereas our task requires us to include information from ground truth and previous proof steps.

Theory Solvers. SMT solvers like Z3 (de Moura and Bjørner, 2008) and CVC (Barbosa et al., 2022) are commonly used to check the satisfiability and validity of logical formulas. They have enabled applications like system verification, program analysis, and model checking. Given a set of logical formulas, an SMT solver determines their satisfiability by applying theories and inference rules. Validity can be checked by taking the negation of the formula and testing if the negation is unsatisfiable.

Olausson et al. (2023) have shown that theory solvers can be employed for logical reasoning with natural language. We enhance their methodology by creating an advanced parser that converts natural language to first-order logic, which is more adept at processing naturalistic, real-world data and capable of managing tasks with ambiguous premises and conclusions. We also develop a method to incorporate real-world context (ground truth) into the logical formula.

3 Methodology

3.1 Task Formulation and Background

Our methodology can be used to detect logical fallacies. The input to the task is a natural language sentence, or a set of sentences, that contains an implication (inference), which would be optionally backed by one or more claims. Our methodology processes this input using LLMs and SMT solvers to output if the given input is a logical fallacy or not, and if it is, then produce a natural language counter-example that explains why it is a logical fallacy.

For the task, we introduce some basic background in first-order logic. In first-order logic, propositions are represented using predicates, which express properties or relations over objects in a domain. These predicates can be combined with constants, representing specific objects, and variables, standing for unspecified elements in the domain. Interpretations assign meaning to these symbols within a given context, while sorts categorize objects into different types, facilitating precise reasoning about their properties.

The logical connectives of first-order logic, including implication ($\Rightarrow$), universal quantifier ($\forall$), existential quantifier ($\exists$), and operators for conjunction/*and* ($\wedge$), disjunction/*or* ($\vee$), and negation/not ($\neg$), allow for the construction of intricate statements. Implication captures conditional relationships. Quantifiers enable assertions over elements in the domain: a universal quantifier indicates that a proposition is true for all elements, whereas the existential quantifier indicates that a proposition is true for some elements in the domain. The other connectives follow their trivial definitions and are used to develop compound and meaningful first-order logical statements.

3.2 Module A: Natural Language to First Order Logic

We devise a technique to efficiently convert a given natural language sentence to logical form. Our methodology is split into multiple steps involving few-shot prompting for LLMs. These steps aim at three major goals. The first goal is to be able to split a sentence into multiple smaller components that can be represented at the first-order logic level. The second goal is to identify the relationships between different sub-components to merge them and develop the logical formula. The third goal is to identify real-world relationships between these sub-components (ground truth) and augment them to the first-order logical formula in order to incorporate context in the statement. We would use two simple examples to explain the algorithm: Example 1 below is a logical fallacy and Example 2 is a valid statement.

> Example 1: I met a tall man who loved to eat cheese, now I believe all tall people like cheese.
> Example 2: A boy is jumping on skateboard in the middle of a red bridge. Thus, the boy does a skateboarding trick.

The first step is to develop a semantic inference module to transform a natural language sentence to claim and implication form. Generally, a sentence can be split into some claims and some implication based upon those claims. It is also possible for a sentence to have no claim, which means that the entire sentence is being asserted with respect to the ground truth, which we evaluate in later steps.

> Example 1: *Claim:* A tall man loved to eat cheese. *Implication*: All tall people like cheese.
> Example 2: *Claim:* A boy is jumping on skateboard in the middle of a red bridge. *Implication:* the boy does a skateboarding trick.

Next, we split the claim and implication into various sub-components. We utilize these sub-components to extract the meaning of the sentence from ground up and eventually build up the logical form of the sentence.

The first set of sub-components are referring expressions. Referring expressions or entities are used to identify specific entities and could be any noun phrase, or surrogate for a noun phrase, whose function in discourse is to identify some objects. Additionally, we find the relationship between different entities using Zero-Shot classification via Natural Language Inference (NLI). These relationships (subset / equality / not related) are generally helpful in adding appropriate quantifiers in the logical form of the sentence. For example, if the entities are ‘man’ and ‘people’, then it can be inferred that ‘man’ is a subset of ‘people’, and that the man would be bound by an existential quantifier in the sentence.

> Example 1: *Referring Expressions:* man: x, cheese: c, people: y, $x \subseteq y$
> Example 2: *Referring Expressions:* boy: x, skateboard: s, bridge, skateboardingTrick: y

The other set of sub-components are properties, which are used describe a trait of a referring expression or a relationship between multiple referring expressions. These properties are essentially predicates in first-order logic. We also find the relationships between numerous properties. For example, in Example 1, it can be inferred that ‘Like’ and ‘Love’ are contextually similar. Similarly, in the valid example, ‘jumping over skateboard’ implies ‘doing a skateboard trick’. These relationships represent a form of ground truth/context that is not directly present in the statement.

To identify these contextual relationships, we run NLI between each pair of properties, i.e, by setting one property as the hypothesis and the other as the premise as the input to the NLI model. If we find that any one property entails the other, we add the relationship $\text{property1} \Rightarrow \text{property2}$ to our context. Before running the NLI model between a pair of properties, we replace the variables in each property with the referring expressions that they represent. This adds additional context that helps the NLI model identify relations. For example, in Example 2, the NLI model is unable to find the relation between *JumpsOn(x,s)* and *Does(x,y)*, but is able to identify the relationship between *JumpsOn(boy,skateboard)* and *Does(boy,skateboardingtrick)*. Without these additional ground truth assertions, we may not be able to prove validity of the statement.

Example 1: *Properties:* Tall, Love, Like
Relationships: $\text{Tall}(x), \text{Love}(x, c)$
Ground truth:
- $\forall x(\text{Like}(x, c) \Rightarrow \text{Love}(x, c))$
- $\forall x(\text{Love}(x, c) \Rightarrow \text{Like}(x, c))$
- $x \subseteq y$

Example 2: *Properties:* JumpsOn, inMiddleOf, Red, Does
Relationships: $\text{JumpsOn}(b, s), \text{Red}(\text{bridge}), \text{inMiddleOf}(b, \text{bridge}), \text{Does}(b, y)$
Ground truth:
- $\forall x(\text{JumpsOn}(x, s) \Rightarrow \text{Does}(x, y))$

Finally, we combine all of the information with the help of an LLM and utilizing the relationships between numerous properties and entities to obtain the first-order logical form of the sentence. For a logical fallacy, the negation of the formula is expected to be satisfiable. For a valid statement, the negation of the formula should be unsatisfiable. This leads us to the next step, which is to feed the formula to an SMT solver.

Example 1: *First-Order Logic:*
$((\forall x(\text{Like}(x, c) \Rightarrow \text{Love}(x, c)))\wedge (\forall x(\text{Love}(x, c) \Rightarrow \text{Like}(x, c)))\wedge\ (\exists x(\text{Tall}(x) \wedge \text{Love}(x, c)))) \Rightarrow (\forall y(\text{Tall}(y) \Rightarrow \text{Like}(y, c)))$
Example 2: *First-Order Logic:*
$((\forall x \text{JumpsOn}(x, s) \Rightarrow \text{Does}(x, y)) \wedge \text{Red}(\text{bridge})\wedge\ \text{inMiddleOf}(b, \text{bridge}) \wedge \text{JumpsOn}(b, s)) \Rightarrow \mathit{Does}(b, y)$

3.3 Module B: FOL to SMT Solving

Our next step involves automatically creating an SMT file for the negation of the first-order logical formula generated. Given a logical formula, while one can easily write an SMT file for the same manually, generating one automatically for an arbitrary formula is something that has not been done before, and is one of our major contributions.

We have developed an efficient compiler for parsing a given logical formula and converting it into a SMT file that can be given as input to CVC, as described in Algorithm 1. This compiler translates any first-order logic formula to the SMT input format, ensuring that no SMT programming is required by future users of this method. Some of the major challenges involved in designing the compiler were in designing a recursive infix to prefix algorithm to parse the input formula, as well as designing a novel algorithm (Algorithm 2, present in the Appendix) to identify and unify sorts.

Algorithm 1 Logical Formula to SMT Compilation

1. Split the formula across any operator, parentheses, or commas into tokens.
2. Process tokens to instances of operators, variables and predicates. For predicates, identify all arguments and recursively process tokens for the arguments separately.
3. Convert the main logical formula from infix to prefix form. For predicates, recursively convert the arguments to prefix form.
4. Identify sorts of all variables and predicates using unify_sort described in Algorithm 2.
5. Parenthesize the prefix form formula to bring it into SMT format appropriately.
6. Create the SMT file by declaring appropriate sorts, variables and predicates using $(\text{declare} - \text{sort})$ and $(\text{declare} - \text{fun})$. Assert negation of the logical formula. Add $(\text{check} - \text{sat})$ and $(\text{get} - \text{model})$ to the SMT file.

3.4 Module C: Interpretation of SMT Solver Results

We send the SMT file that we generate to the cvc4 solver (Barrett et al., 2011) to get the result (sat / unsat), and if it is satisfiable, then get a model, i.e, a concrete assignment of values to the variables in the formulas that makes the formulas true. Since we assert the negation of the actual logical formula, this model acts as a counter-example to the original formula, proving that the given claim and implication is actually a logical fallacy.

Generally, it is difficult to understand the model generated by the SMT solver, especially for a layperson. In order to explain the counter-example better to prove that the reasoning is faulty, it is essential to explain the counter-example in natural language.

A simplified example for the same is given in Fig-

```
I met a tall man who loved to eat cheese, now I believe all
tall people like cheese.

(assert (not (=> (exists ((x BoundSet)) (and (T x) (L x cheese)))
                 (forall ((y BoundSet)) (=> (T y) (L y cheese))))))

sat
(
; cardinality of BoundSet is 2
; rep: (as @BoundSet_0 BoundSet)
; rep: (as @BoundSet_1 BoundSet)
; cardinality of UnboundSet is 1
; rep: (as @UnboundSet_0 UnboundSet)
(define-fun cheese () UnboundSet (as @UnboundSet_0 UnboundSet))
(define-fun T ((_arg_1 BoundSet)) Bool true)
(define-fun L ((_arg_1 BoundSet) (_arg_2 UnboundSet)) Bool (and (=
(as @BoundSet_0 BoundSet) _arg_1) (= (as @UnboundSet_0 UnboundSet)
_arg_2)))
)

BoundSet = Set of Tall People.
Let @BoundSet_0 = John (tall person 1) and @BoundSet_1 = Jane
(tall person 2)
UnboundSet = Set of Foods. Let @UnboundSet_0 = Cheese
Predicates: T(person) = person is tall.  L(person, food) = person
likes food
SMT Instance:
•John is tall (T(John) is True). John likes cheese (L(John,
 Cheese) is True).
•Jane is tall (T(Jane) is True). No constraint Jane likes cheese.
Therefore, there exists a tall person (John) who likes cheese, but
it does not follow that all tall people like cheese, since Jane
serves as a counter-example.
```

Figure 2: Interpretation of results from a counter-example.

ure 2. As evident, the SMT result is hard to understand because it uses technical terminology that can generally be only understood by those who understand cvc4 and SMT well. Therefore, we developed a pipeline to convert the cvc4 results back to natural language to explain why the reasoning is faulty.

To do so, we prompt an LLM and give it the claim, implication, referring expressions, properties, first-order logical formula and the counter-model generated by cvc4. The transformer model is then utilized to interpret the counter-model using natural language as depicted in Figure 2.

4 Experimental Results

4.1 Dataset

We use the following three datasets to evaluate the effectiveness of our approach:

- LOGIC (Jin et al., 2022): consists of 2,449 common logical fallacies.
- LOGICCLIMATE (Jin et al., 2022): consists of 1,079 logical fallacies from climate change news from the Climate Feedback website.
- Stanford Natural Language Inference (SNLI) Corpus (Bowman et al., 2015): contains over 170,000 valid sentences generated by combining 'sentence 1' and 'sentence 2' from the entailment data points to form a sentence where claim entails the implication.

Since the LOGIC and LOGICCLIMATE datasets contain only logical fallacies, we randomly sample equal number of valid statements from the SNLI corpus to balance the datasets.

4.2 Models

We compare our method with NLI Zero Shot Classifiers (BART MNLI) and pretrained language models, including Llama-7b, Mistral-7b (Jiang et al., 2023), GPT3.5 (OpenAI, 2024), GPT4 (OpenAI et al., 2024) and Claude-3-Opus (Anthropic, 2024) with few-shot in-context examples. We use the open-source model Llama 2 (7B-parameters) (Touvron et al., 2023) for LLM prompting and BART (140M parameters) (Lewis et al., 2020) finetuned on MNLI (Williams et al., 2018) for identifying the relationships between properties and referring expressions. We run the experiments on a V100 GPU, and one run costs around 2 GPU hours.

4.3 Main Results

Model	Acc	P	R	F1
BART-MNLI (Zero Shot)	0.58	**1**	0.15	0.26
Llama-7b (Few Shot)	0.41	0.45	0.82	0.58
Mistral-7b-Instruct (Few Shot)	0.85	0.85	0.86	0.85
GPT3.5 (Few Shot)	0.88	0.86	0.91	0.89
GPT4 (Few Shot)	0.95	0.97	0.94	0.95
GPT4 (Few Shot with COT)	0.94	0.95	0.94	0.94
Claude 3 Opus (Few Shot)	**0.97**	0.96	**0.98**	**0.97**
NL2FOL (Few Shot)	0.63	0.58	0.92	0.71

Table 2: Model performance on the LOGIC+SNLI dataset, showcasing accuracy (Acc), precision (P), recall (R), and F1 score (F1).

As shown in the experimental results in Table 2, we find that LLMs can effectively identify properties and referring expressions in the sentence, and natural language inference can be used to identify the relationships between properties and entities well.

We observe that our method achieves an F1-Score of 71%, surpassing both end-to-end few-shot and zero-shot classification techniques with the same models. When used end-to-end, the Llama-7b model reached only a 58% F1-Score, while zero-shot NLI classification with the BART-MNLI model was ineffective, incorrectly labeling every sentence as a logical fallacy. Although other language models have shown better performance, comparisons may be skewed as these models might have been exposed to the LOGIC dataset and its labels during their training, as this dataset was compiled from publicly accessible web sources. Our model demonstrated high recall in identifying logical fallacies, suggesting it is well-suited for detecting and addressing misinformation.

4.4 Results on the Challenge Set

Our challenge set, LOGICCLIMATE+SNLI, is a set of real-world logical fallacies from climate change news. The results obtained are shown in Table 3. As we are using this dataset to test generalization, the in context examples we provide to all the models are from the LOGIC dataset. Our methodology leads to results that are highly similar to the results of the LOGIC dataset. This demonstrates that our system is exceptionally robust and adapts well to real-world text, including texts with significant domain-specific context. This makes it highly effective in detecting and mitigating misinformation. This dataset is more of a fair comparison, as it is unlikely that these models have seen the dataset during training as the data is human-annotated. We find our model outperforms all LLMs we test on, despite being much smaller. The LLMs achieve a high precision but low recall, indicating that they can classify the valid sentences from SNLI effectively, but not the fallacies from the LOGICCLIMATE dataset.

Metric	**Acc**	**P**	**R**	**F1**
BART-MNLI (Zero Shot)	0.57	**1**	0.14	0.25
Llama-7b (Few Shot)	0.31	0.38	0.62	0.47
Mistral-7b-Instruct (Few Shot)	0.62	0.68	0.44	0.53
GPT3.5 (Few Shot)	0.63	0.81	0.39	0.53
GPT4 (Few Shot)	0.64	0.91	0.30	0.45
GPT4 (Few Shot with COT)	0.66	0.90	0.36	0.51
Claude3 Opus (Few Shot)	**0.67**	0.92	0.38	0.54
NL2FOL (Few Shot)	0.66	0.60	**0.94**	**0.73**

Table 3: Comparison of accuracy (Acc), precision (P), recall (R), and F1 score (F1) Metrics for various approaches for the LOGICCLIMATE+SNLI dataset.

4.5 Error Analysis

As evident from the results, proving a statement to be valid is harder than identifying it as a logical fallacy, contributing to the lower precision of the model. This is because it is inherently difficult to prove the negation of a statement as unsatisfiable compared to satisfiable. This challenge arises because the model may not have articulated some semantics or ground truth in the first-order logical formula that may be necessary to prove validity. If this context is not well established in the SMT code explicitly, we cannot prove validity, because it would be easy to build a counter-example. The SMT needs full context, and any gaps in contextual information can cause a valid statement to be mistakenly identified as a logical fallacy.

One such case is present in example 4 of the Table 4. In this case, the model is not able to identify the extra context statement because the NLI model does not identify a required ground-truth relation. If this context were identified and added to the claim of the logical formula, then it would have predicted the statement to be valid.

Furthermore, our current approach is limited to discerning the NLI relationship between two properties at a time, rather than handling multiple relationships concurrently. For example, consider example 6 in Table 4. In the given example, the semantic claim involves the conjunction of two properties entailing the third, while the NLI model only checks if one property entails the other. Finding such complex extra context requires more advanced techniques or human intervention, and including this can further improve the precision of the model.

Nonetheless, it is important to clarify that these examples do not imply a general inefficiency of NLI. An interesting illustration of where they work well can be found in Example 5 from Table 4. In this instance, our model identifies additional context by establishing relationships such as IsBaseballPlayer implying IsPlayingBaseball, and IsNearOutfieldFence implying IsOutdoors. These contextual connections help in effectively proving the validity of the statement. Note that if we did not have these semantic relations embedded, the predicates won't have a meaning attached to them and could be assigned any invalidating interpretation. This substantiates the need of including semantic information in the formula.

The examples in Table 4 prove that few shot prompting with LLMs is a great way to convert a sentence to first-order logic in a step-by-step manner, improving the efficiency of conversion. It can be seen that most of these examples identify referring expressions and properties quite well, and are able to produce syntactically correct expressions. Various examples like example 1 and 2 correctly detect logical fallacies. Consequently, we also obtained a very high recall for NL2FOL, beating the baseline techniques by a significant amount.

	Type	Sentence	Logical Form	Prediction
1	LF	X has been around for years now. Y is new. Therefore, Y is better than X.	(IsNew(a) ∧ ∼ IsNew(b)) ⇒ (IsBetterThan(a,b))	LF: Correct prediction
2	LF	Jimmy isn't at school today. He must be on a family trip.	(∼IsAtSchool(a)) ⇒(IsOnFamilyTrip(a))	LF: Correct prediction
3	LF	Everyone is doing the Low-Carb Diet.	(∃ b (∃ a (IsDoing(b,a)))) ⇒ (∃ c (∃ a (IsDoing(c,a))))	Valid: Incorrect prediction: Wrong translation when there was no claim given
4	V	Two dogs are fighting in a field. Consequently, the two dogs are outside.	(∃ b (∃ a (IsFighting(a, b) ∧ IsInField(b) ∧ IsInField(b)))) ⇒ (∃ a (IsOutside(a)))	LF: Incorrect prediction: Missing semantic ground truth claim: ∀ a (IsInField(a) ⇒ IsOutside(a))
5	V	A baseball player gets ready to catch a fly ball near the outfield fence. Therefore, a person is playing baseball outdoors.	(∃ a (IsGettingReady(a) ∧ (IsABaseballPlayer(a) ∧ IsCatchingFlyBall(a) ∧ IsNearOutfieldFence(a))) ∧ (∀ e (IsABaseballPlayer(e) ⇒ IsPlayingBaseball(e))) ∧ (∀ f (IsPlayingBaseball(f) ⇒ IsABaseballPlayer(f))) ∧ (∀ g (IsNearOutfieldFence(g) ⇒ IsOutdoors(g)))) ⇒ (∃ c (∃ a (IsPlayingBaseball(a) ∧ IsOutdoors(c))))	Valid: Correct Prediction
6	V	A woman sits alone on a park bench in the sun. Hence, a women is in a park.	(IsSittingOn(a, b) ∧ isParkBench(b) ∧ IsInSun(a)) ⇒ (IsInPark(a))	LF: Incorrect prediction: Missing semantic ground truth claim: $\forall a \forall b$ (IsSittingOn(a, b) ∧ isParkBench(b) ⇒ IsInPark(a))

Table 4: Some example outputs of our model. Type *LF* refers to *Logical Fallacy* and *V* refers to *Valid* statement.

Among the few logical fallacies where our model incorrectly predicted a logical fallacy to be a valid statement, most of these predictions failed due to the imprecision of the LLM, leading to false translations and incorrect results. Example 3 is a prominent case where the input does not have any claim, rather just jumps to an implication. However, the model is not able to identify that the example has no claim. As a result, we get an incorrect translation from our technique. We believe that utilizing more advanced LLMs in future experiments will help prevent these issues and improve our statistics further.

5 Future Work

Potential approaches to improving performance on this task include utilizing more advanced LLMs, utilizing DSPy (Khattab et al., 2023) to optimize prompts, utilizing Constrained Decoders (Geng et al., 2024) to ensure the generated output follows the correct syntax or utilizing self-consistency (Wang et al., 2023) to verify the method's intermediate outputs. The step-by-step, interpretable nature of our approach also enables the incorporation of human feedback into the pipeline in the future.

As there are currently no large datasets containing natural language formulas with annotated first-order logical forms, we encourage researchers to utilize our method to generate psuedo first-order logic labels to fine-tune models. Having demonstrated an effective method to compile, execute, and verify the validity of logical formulas, this direction could be used to develop benchmarks for logical reasoning tasks in NLP, that are evaluated similarly to how code generation benchmarks are evaluated by compiling the generated code and running unit tests (Jimenez et al., 2024).

6 Conclusion

In conclusion, we presented an automatic and effective solution for detecting fallacies and tackling misinformation. We developed a strategy to distinguish logical fallacies from valid statements, which involves a chaining approach to convert a sentence to first-order logic using LLMs, followed by using SMT solvers to identify whether the first-order logical statement is valid or not, and if not, interpret the counter-model generated by the SMT solver in natural language. Our proposed technique showed promising results in identifying logical fallacies and valid statements, as well as great generalization ability. The primary bottleneck is the natural language to first-order logic conversion, which is ongoing research.

Limitations

The step-by-step nature of our model increases the time taken for inference compared to end-to-end models. Moreover, it also increases the development time as a user needs to write prompts for each step.

As discussed in the analysis section, correct identification of ground truth knowledge is crucial for our method. At the moment, our method only considers simple relations between properties (ex: $a \Rightarrow b$) and misses out on complex relations (ex: $(a \wedge b) \Rightarrow (c \vee d)$))

While, we expect the technique to generalize to datasets in languages other than English and models other than LLAMA-7b and BART-MNLI, testing this is left to future work.

Ethics Statement

While the intended outcome of this research is to help fight misinformation and promote rational discourse, there are several ethical challenges that we must consider. Dependence on AI for identifying logical fallacies could influence how individuals engage in debates and discussions. There's a risk that people may over-rely on AI judgments, potentially stifling complex arguments or dissenting opinions that are essential for a healthy democratic process. The use of AI in moderating discussions, especially in identifying logical fallacies, raises ethical questions about the automation of content moderation. While it can enhance the quality of public discourse by filtering out fallacious arguments, it also risks automating censorship and impacting the dynamics of online communities. In the wrong hands, logical fallacy detection tools could be used to censor speech or suppress certain viewpoints under the guise of promoting rational discourse. Governments or organizations might misuse these tools to silence opposition or critique, posing a threat to free speech and open debate.

To mitigate these issues, there is a need to establish ethical guidelines for the use of AI in public discourse, including transparency, accountability, and user engagement. It is necessary to encourage public literacy in AI and logical fallacies, empowering individuals to critically assess both AI judgments and arguments in discussions.

References

Tariq Alhindi, Tuhin Chakrabarty, Elena Musi, and Smaranda Muresan. 2022. Multitask instruction-based prompting for fallacy recognition. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8172–8187, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Rami Aly, Marek Strong, and Andreas Vlachos. 2023. Qa-natver: Question answering for natural logic-based fact verification.

Gabor Angeli and Christopher D Manning. 2014. Naturalli: Natural logic inference for common sense reasoning. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 534–545.

Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku.

H. Barbosa et al. 2022. cvc5: A versatile and industrial-strength smt solver. In *Tools and Algorithms for the Construction and Analysis of Systems. TACAS 2022. Lecture Notes in Computer Science*, volume 13243, Cham. Springer.

Dave Barker-Plummer, Richard Cox, and Robert Dale. 2009. Dimensions of difficulty in translating natural language into first-order logic. In *International Working Group on Educational Data Mining, Paper presented at the International Conference on Educational Data Mining (EDM) (2nd, Cordoba, Spain, Jul 1-3, 2009)*, pages 220–229.

Clark Barrett, Christopher L. Conway, Morgan Deters, Liana Hadarean, Dejan Jovanović, Tim King, Andrew Reynolds, and Cesare Tinelli. 2011. Cvc4. In *Computer Aided Verification*, pages 171–177, Berlin, Heidelberg. Springer Berlin Heidelberg.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

M. Cosler, C. Hahn, D. Mendoza, F. Schmitt, and C. Trippel. 2023. nl2spec: Interactively translating unstructured natural language to temporal logics with large language models. In *Computer Aided Verification. CAV 2023. Lecture Notes in Computer Science*, volume 13965, Cham. Springer.

Leonardo de Moura and Nikolaj Bjørner. 2008. Z3: An efficient smt solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340, Berlin, Heidelberg. Springer Berlin Heidelberg.

Francesco Fuggitti and Tathagata Chakraborti. 2023. NL2LTL – a python package for converting natural language (NL) instructions to linear temporal logic (LTL) formulas. In *AAAI*. System Demonstration.

Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. 2024. Grammar-constrained decoding for structured nlp tasks without finetuning.

Ivan Habernal et al. 2018. Before name-calling: Dynamics and triggers of ad hominem fallacies in web argumentation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 386–396, New Orleans, Louisiana. Association for Computational Linguistics.

Christopher Hahn, Frederik Schmitt, Julia J Tillman, Niklas Metzger, Julian Siber, and Bernd Finkbeiner. 2022. Formal specifications from natural language. *arXiv preprint arXiv:2206.01962*.

C. L. Hamblin. 2022. *Fallacies*. ADVANCED REASONING FORUM.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. Swe-bench: Can language models resolve real-world github issues?

Zhijing Jin, Abhinav Lalwani, Tejas Vaidhya, Xiaoyu Shen, Yiwen Ding, Zhiheng Lyu, Mrinmaya Sachan, Rada Mihalcea, and Bernhard Schoelkopf. 2022. Logical fallacy detection. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 7180–7198, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2023. Dspy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*.

M. Lewis et al. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 2020 Association for Computational Linguistics (ACL)*.

J.X. Liu, Z. Yang, B. Schornstein, S. Liang, I. Idrees, S. Tellex, and A. Shah. 2022. Lang2ltl: Translating natural language commands to temporal specification with large language models. In *Workshop on Language and Robotics at CoRL 2022*.

Bill MacCartney and Christopher D Manning. 2014. Natural logic and natural language inference. In *Computing Meaning: Volume 4*, pages 129–147. Springer.

E. Musi et al. 2022. Developing fake news immunity: fallacies as misinformation triggers during the pandemic. *Online Journal of Communication and Media Technologies*, 12(3):e202217.

Callistus Ireneous Nakpih and Simone Santini. 2020. Automated discovery of logical fallacies in legal argumentation. *International Journal of Artificial Intelligence and Applications (IJAIA)*, 11.

Theo Olausson, Alex Gu, Ben Lipkin, Cedegao Zhang, Armando Solar-Lezama, Joshua Tenenbaum, and Roger Levy. 2023. LINC: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5153–5176, Singapore. Association for Computational Linguistics.

OpenAI. 2024. Openai gpt-3 api [gpt-3.5-turbo].

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, and Shyamal Anadkat et al. 2024. Gpt-4 technical report.

William C Purdy. 1991. A logic for natural language. *Notre Dame Journal of Formal Logic*, 32(3):409–425.

Emily Sheng et al. 2021. "nice try, kiddo": Investigating ad hominems in dialogue responses. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 750–767, Online. Association for Computational Linguistics.

Hrituraj Singh, Milan Aggrawal, and Balaji Krishnamurthy. 2020. Exploring neural models for parsing natural language into first-order logic.

Zhivar Sourati, Vishnu Priya Prasanna Venkatesh, Darshan Deshpande, Himanshu Rawlani, Filip Ilievski, Hông Ân Sandlin, and Alain Mermoud. 2023. Robust and explainable identification of logical fallacies in natural language arguments.

Christian Stab and Iryna Gurevych. 2017. Recognizing insufficiently supported arguments in argumentative essays. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 980–990, Valencia, Spain. Association for Computational Linguistics.

Christopher W. Tindale. 2007. *Fallacies and Argument Appraisal*. Critical Reasoning and Argumentation. Cambridge University Press.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, and Amjad Almahairi et al. 2023. Llama 2: Open foundation and fine-tuned chat models.

Eemeren F H van and Snoeck Henkemans Arnolda Francisca. 2017. *Argumentation: Analysis and evaluation*. Routledge.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models.

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

Yuan Yang, Siheng Xiong, Ali Payani, Ehsan Shareghi, and Faramarz Fekri. 2023. Harnessing the Power of Large Language Models for Natural Language to First-Order Logic Translation. *arXiv e-prints*, page arXiv:2305.15541.

A Unify Sort Algorithm

Algorithm 2 unify_sort for predicate, say $A(x, y)$

1. Declare the current sort of A: (NULL, NULL, Bool)
2. For each instance of predicate A:
   (a) Find the sort of arguments based upon the instance (instance sort):
      i. If argument is a formula, then sort(arg) = Bool.
      ii. If argument is a variable, then sort(arg) = sort(variable) *[may be null]*
   (b) Unify current sort with instance sort:
      i. If sorts of an argument in the current sort and instance sort are not NULL and different, then raise Error (incompatible sorts).
      ii. If current argument sort is NULL and corresponding instance sort is not NULL, then update current argument sort = instance sort.
      iii. If instance argument sort is NULL and corresponding current sort is not NULL, then update the sort of the corresponding variable to current sort.