Formalizing Simple Natural Language Arguments using Abstract Meaning Representation and Approximate Propositional Reasoning

Xuyao Feng
Department of Computer Science
University College London, United Kingdom
xuyao.feng.20@ucl.ac.uk

Anthony Hunter
Department of Computer Science
University College London, United Kingdom
anthony.hunter@ucl.ac.uk

***Abstract*—Argumentation is an important cognitive activity that involves scrutinizing arguments and counterarguments. Ideally, an argument's premises should entail its claim while maintaining consistency between the premises and the claim. Furthermore, for a pair of arguments, it is important to determine whether one supports or attacks the other. Argument mining is being developed to automatically extract arguments from text, and identify support or attack relationships between them. But there would be advantages of using a formal logic representation of the arguments as it may offer a clearer and less ambiguous representation of information, and it would support automated reasoning. Whilst there are some frameworks for modeling logical argumentation, there is a lack of methods to translate text from argument mining into logical formulas. To address this gap, we propose a neuro-symbolic pipeline combining a pre-trained large language model (LLM) with neuro-symbolic reasoning. This converts free-text premises and claims such as from argument mining into logical formulas, based on abstract meaning representation (AMR), and employs a SAT solver for entailment and contradiction checks. We apply this method to the Sentences Involving Compositional Knowledge (SICK) and STS-B (Semantic Textual Similarity Benchmark) datasets, and provide promising performance results. We then explain how our pipeline can be used construct structured argument graphs from simple natural language arguments.**

***Index Terms*—Argumentation, Commonsense Reasoning, Argument graphs**

I. Introduction

Humans often resort to argumentation when faced with conflicting evidence and opinions [1]. Argument mining [2], a form of applied natural language processing, involves detecting arguments in text [3], identifying the claims and premises of those arguments [4], and identifying relations such as entailment, support, or conflict [5]. However, argument mining does not provide formal logical formulas for the extracted premises and claims, nor does it show how they may form a logical argument. Yet, considering arguments in terms of logical formulas allows for a more precise representation of the information in arguments, and it allows for the use of automated reasoning to explain how premises entail (or not entail) a claim, and to explain how one argument attacks or supports (or does not) another. There are formalisms that provide a formal language for constructing arguments, where the premises and claims are explicit, and their relationship is formally defined [6] (e.g. ABA [7], ASPIC+ [8], and deductive argumentation [9]). However, there is no systematic method to translate the text from argument mining into logical arguments and use automated reasoning based on these frameworks.

First sentence (S1)	Second sentence (S2)	Label
A man is patting an alligator on the mouth.	An alligator is being patted on the mouth by a man.	Ent
A group of people is equipped with gear used for protection.	A group of people is equipped with protective gear.	Ent
The crowd is watching the football at the game.	A group of football players is running in the field.	Neu
There is no boy in a white t-shirt splashing in shallow water.	A boy in a white t-shirt is splashing in shallow water.	Con

TABLE I: Some examples from STS-B and SICK datasets where Ent means S1 entails S2, Con means S1 contradicts S2, and Neu means that S1 neither entails nor contradicts S2.

To address this shortcoming, we want to translate sentences such as in Table I into logic so that we can use automated reasoning to identify whether or not the first sentence entails or contradicts the second. For this, we propose a neuro-symbolic pipeline (summarized in Figure 1) that integrates a pre-trained large language model (LLM) to translate sentences of text (e.g. output from argument mining) into logical formulas (via abstract meaning representation), and automated reasoning, based on a SAT solver, to determine whether their corresponding logical formulas are such that the relationship of one to the other is entailment, contradiction, or neutral (i.e. neither entailment nor contradiction). In order to address the problem that the sentences may assume some implicit information, our automated reasoning incorporates relaxation methods for treating two logical atoms as being the same if their word embeddings are sufficiently similar, and for ignoring some atoms under certain conditions. These relaxation methods capture a form of commonsense reasoning capability.

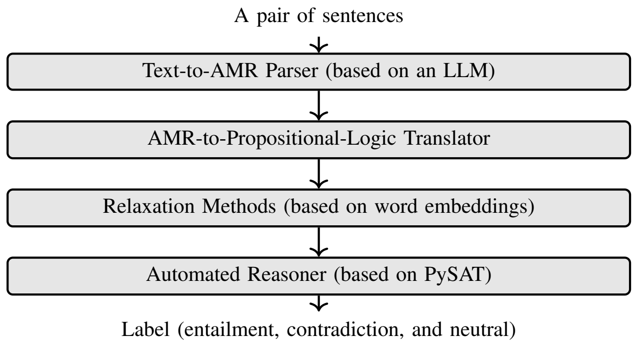
A pair of sentences
Text-to-AMR Parser (based on an LLM)
AMR-to-Propositional-Logic Translator
Relaxation Methods (based on word embeddings)
Automated Reasoner (based on PySAT)
Label (entailment, contradiction, and neutral)

Fig. 1: Our neuro-symbolic pipeline. The input is a pair of sentences (e.g. a premise and claim), and the output is a label.

To evaluate our neuro-symbolic pipeline, we use the STS-B (Semantic Textual Similarity Benchmark) and the SICK (Sentences Involving Compositional Knowledge) datasets. The STS-B dataset is a set of pairs of sentences and an annotation of whether the first entails (or not) the second. The SICK dataset is a set of pairs of sentences and an annotation of whether the relation of the first to the second is entailment, contradiction, or neutral. We give some examples in Table I. Our experiments demonstrate that the neuro-symbolic pipeline achieves promising performance across standard evaluation metrics, including precision, recall, F_1-score, and accuracy.

We conclude the paper with a discussion of how our neuro-symbolic pipeline can be used to generate a structured argument graph where each node is either a premise or a claim, and each arc is either a form of support (entailment) or a form of attack (contradiction). This allows us to have a clear visualization of how premises support claims and how arguments support or contradict each other.

II. Background

This section reviews some existing methods, namely abstract meaning representation (AMR), and automated reasoning based on SAT solvers, that we incorporate in our pipeline.

A. Abstract meaning representation

Abstract meaning representation (AMR) is a semantic representation language for representing sentences as rooted, labelled, directed, and acyclic graphs (DAGs). AMR is intended to assign the same AMR graph to similar sentences, even if they are not identically worded. The approach was first introduced by Langkilde and Knight in 1998 [10] as a derivation from the Penman Sentence Plan Language [11]. In 2013, AMRs re-gained attention due to Banarescu et al. [12], and were introduced into NLP tasks such as machine translation and natural language understanding. The modern (post-2010) AMR[1] draws on predicate senses and semantic roles from the OntoNotes project [13].

In AMR, negation is represented via the :polarity relation. For example, Figure 2 represents "The boy does not want to go." In AMR, the numbers after the instance name (such as want-01 above) denote a particular OntoNotes or PropBank semantic frame [14]. These frames have different parameters, but the subject is generally denoted by `arg0` and the object by `arg1`. The parameters, which draw out the semantic roles of the words in the AMR, include `location` (e.g. "France"), `unit` (e.g. "kilogrammes"), and `time` (e.g. "yesterday"). AMR uses the same structure to represent semantically similar texts by making several simplifying assumptions. AMR cannot represent verb tenses nor distinguish between verbs and nouns. Also it does not represent articles, quote marks, or the singular and plural. Nonetheless, AMR offers a valuable formal abstraction of the meaning of sentences.

```
(w / want-01                 (w / want-01
      :arg0 (b / boy)              :arg0 (b / boy)
      :arg1 (g / go-01             :arg1 (g / go-01
            :arg0 b))                    :arg0 b
                                         :polarity -))
```

Fig. 2: AMR for the sentence "The boy wants to go." (left) and "The boy does not want to go." (right).

In our pipeline, we use the IBM Transition AMR parser[2] to load the pre-trained ensemble AMR 3.0 model (AMR3-joint-ontowiki-seed43), which combines Smatch-based ensembling techniques with ensemble distillation [15]. This parser translates a sentence of text into an AMR graph.

B. Automated reasoning

One of the advantages of AMR is that we can easily transform an AMR graph into first-order logic formulas using the Bos algorithm [16] which translates each graph into a nested conjunction of atoms, where each monadic atom is a concept and each dyadic atom is relation between a pair of concepts [17]. An example of this is shown below, where the AMR for "The boy does not want to go" from Figure 2 is converted into a logical formula as follows.

$$\exists \mathtt{w}(\exists \mathtt{b}(\mathtt{want}(\mathtt{w}) \land \mathtt{arg0}(\mathtt{w},\mathtt{b}) \land \mathtt{boy}(\mathtt{b}) \land \neg\exists \mathtt{g}(\mathtt{arg1}(\mathtt{w},\mathtt{g}) \land \mathtt{go}(\mathtt{g}) \land \mathtt{arg0}(\mathtt{g},\mathtt{b}))))$$

There is an open-source Python library based on the Bos algorithm, the AMR-to-logic converter [17], to translate AMR graphs into first-order logic. We extended this library, to give our AMR-to-propositional-logic translator, by rewriting each first-order logic formula to a propositional logic formula by grounding out the existentially quantified variables with new constants. We refer to each such propositional formula as an AMR formula. For this, we make the assumption that for each existentially quantified variable, there is a specific entity that can represent the quantified variable. This can be viewed as a Skolem constant. We choose each constant symbol for this grounding as follows: For each monadic predicate $r(a)$, we use r as the constant symbol to replace the

[1]https://github.com/amrisi/amr-guidelines

[2]https://github.com/IBM/transition-amr-parser/tree/master

variable symbol a throughout the formula. This then means the monadic predicates are now redundant and so we delete them. Therefore, for the above example, the propositional logic formula is simplified as follows.

$$\texttt{arg0}(\texttt{want}, \texttt{boy}) \land \neg(\texttt{arg1}(\texttt{want}, \texttt{go}) \land \texttt{arg0}(\texttt{go}, \texttt{boy}))$$

For our pipeline, we assume the usual definitions for propositional logic. We start with a set of propositional atoms (letters), and we constructed formula in the usual way using the connectives for negation $\neg$, conjunction $\land$, disjunction $\lor$, implication $\leftarrow$, and biconditio1nal $\leftrightarrow$. For the automated reasoning, we transform all propositional logic formulas into conjunctive normal form (CNF) using SymPY [18]: CNF is defined in the usual way: (1) A literal is either a propositional variable, or the negation of one; (2) A clause is a disjunction of literals; and (3) A formula in conjunctive normal form (CNF) if it is a literal, or a clause, or a conjunction of clauses.

In our pipeline, we use PySAT [19], which integrates several widely used state-of-the-art SAT solvers as theorem provers to check whether a CNF is consistent. Finding an interpretation that fulfils a given Boolean formula is known as the Boolean satisfiability problem (SAT). For example, the formula $\texttt{a} \land \neg\texttt{b}$ is satisfiable as $\texttt{a}$ = TRUE and $\texttt{b}$ = FALSE resulting in the formula being TRUE. In contrast, $\texttt{a} \land \neg\texttt{a}$ is unsatisfiable.

To prove whether a formula ϕ entails a formula ψ, we need to determine whether $\{\phi\} \vdash \psi$ holds. This is equal to determining whether $\phi \land \neg\psi$ is inconsistent. To do this, we need to change $\phi \land \neg\psi$ into a CNF formula, and then we can directly use PySAT to check consistency.

Similarly, to prove whether a formula ϕ contradicts a formula ψ, we need to determine whether $\{\phi, \psi\} \vdash\perp$ holds. To do this, we need to change $\phi \land \psi$ into a CNF formula, and then we can directly use PySAT to check consistency.

III. Method

Our neuro-symbolic pipeline [3] consists of four main components: A text to AMR parser (as described in Section II-A); An AMR-to-propositional-logic translator (as described in section II-B); A set of methods for relaxation in the formulas (which we describe below); And an automated reasoner based on PySAT (as described in section II-B). This pipeline is summarized in Figure 1.

In this section, we explain how we represent the AMR formulas that come from the AMR-to-propositional-logic translator, how we rewrite AMR formulas to abstract formulas for use by a SAT solver, and how we use relaxation methods to simplify these abstract formulas when determining whether entailment or contradiction holds between pairs of formulas.

A. Representing AMR formulas and abstract formulas

An AMR formula is composed from a set of dyadic atoms together with the $\neg$ and $\land$ logical operators. Since each atom is ground, an AMR formula is a propositional logic formula.

[3]https://github.com/fxy-1117/ICTAI2025

Definition 1. *Let $\mathcal{A}$ be a set of dyadic atoms. The set of* **AMR formulas***, denoted $\mathcal{L}$, is defined inductively as follows: If $\alpha \in \mathcal{A}$, then $\alpha \in \mathcal{L}$; If $\alpha, \beta \in \mathcal{L}$, then $\alpha \land \beta \in \mathcal{L}$; And if $\alpha \in \mathcal{L}$, then $\neg\alpha \in \mathcal{L}$.*

Given a sentence S, our AMR-to-propositional-logic translator identifies an AMR formula ϕ that **represents** S.

Example 1. *The AMR formula $\neg\texttt{arg0}(\texttt{go}, \texttt{car})$ represents the sentence "the car does not go".*

An abstract formula is composed from a set of propositional letters together with the $\neg$ and $\land$ logical operators. The set of abstract formulas is also a subset of the set of propositional formulas.

Definition 2. *Let $\mathcal{P}$ be a set of propositional letters. The set of* **abstract formulas***, denoted $\mathcal{F}$, is defined inductively as follows: If $\alpha \in \mathcal{P}$, then $\alpha \in \mathcal{F}$; If $\alpha, \beta \in \mathcal{F}$, then $\alpha \land \beta \in \mathcal{F}$; And if $\alpha \in \mathcal{F}$, then $\neg\alpha \in \mathcal{F}$.*

If $|\mathcal{A}| = |\mathcal{P}|$, then for each $\alpha \in \mathcal{L}$, there is a $\beta \in \mathcal{F}$ (and vice versa) such that α and β are isomorphic (i.e. they have the same syntax tree except for the atoms associated with the leaves). For example, $\neg\texttt{x}_1$ is an abstract formula that is isomorphic to the AMR formula in Example 1.

B. Similarity measures

In order to consider how we can translate AMR formulas into abstract formulas, we need to consider when two atoms can be regarded as equivalent, and therefore mapped to the same propositional letter. For this, if for two AMR atoms α and β, the similarity between two strings of words corresponding to the two atoms is greater than a threshold τ (as explained below), we treat them as equivalent (denoted $\alpha \simeq \beta$).

An **embedding** of a string of words is a vector v obtained by an injective mapping function f. In our pipeline, the f function is the sentence transformer BAAI general embedding model bge-small-en-v1.5 [20] (alternatives can easily be used) that encodes a string of words as a high dimension vector.

Definition 3. *The* **similarity** *between two embedding vectors v_1 and v_2 is defined as follows, where θ is the angle between the vectors, $v_1 \cdot v_2$ is is dot product between v_1, v_2, and $||v_1||$ represents the L2 norm.*

$$\mathsf{similarity}(v_1, v_2) = cos(\theta) = \frac{v_1 \cdot v_2}{||v_1||||v_2||}$$

To obtain a string of words corresponding to an AMR atom, we created a set of 29 templates. Each template is based on taking the information in a dyadic atom in an AMR formula, and representing that information as a natural language sentence. Each template is designed for a type of dyadic atom (i.e. for the predicate name of the atom). For example, for the dyadic atom $\texttt{arg0}(\texttt{play}, \texttt{man})$, the predicate name is $\texttt{arg0}$, and so can use a template for $\texttt{arg0}$ such as "[Y] is the agent performing action [X]" which is a string where [X] and [Y] are placeholders for the first and second arguments of the atom (i.e. $\texttt{play}$ and $\texttt{man}$ in this example).

So from this template, we can instantiate it using the dyadic atom to get "man is the agent performing action play." as the natural language sentence to be used for the word embedding. In order to do this, we require the following definition.

Definition 4. *Let ϕ be an AMR formula, and let $r(a, b)$ be a dyadic atom in ϕ (i.e. $r(a, b) \in \mathsf{Atoms}(\phi)$). Also let T be a template for r with placeholders* [X] *and* [Y]. *The* **instantiate function**, *denoted* Inst, *is defined as follows:* $\mathsf{Inst}(r(a, b), T)$ *= I, where I is the instantiation of T in which* [X] *is replaced by a and* [Y] *is replaced by b.*

Some examples of templates are below where the AMR predicate name (i.e. AMR parameter) is given in brackets.

- (`purpose`) "[Y] is the purpose of action [X]."
- (`time`) "[Y] is when action [X] takes place."
- (`arg1`) "[Y] is the object involved in action [X].".

Next, we use the function f to obtain the sentence embeddings of the instantiations of templates.

Example 2. *Let T = "*[Y] *is the agent performing action* [X]." *be the template for AMR predicate name* $\mathtt{arg0}$. *Therefore, for the AMR atom* $\mathtt{arg0}(\mathtt{play}, \mathtt{child})$, $\mathsf{Inst}(\mathtt{arg0}(\mathtt{play}, \mathtt{child}), T)$ = *"child is the agent performing action play.".*

For the following definition of a matching relation, we assume that one AMR formula refers to a premise and the other refers to a claim.

Definition 5. *Let AMR formula ϕ be a premise and AMR formula ψ be a claim, let $\alpha \in \mathsf{Atoms}(\psi)$ be a dyadic predicate of the form $r(a, b)$, let T_1 be a template for r, let $\beta \in \mathsf{Atoms}(\phi)$ be a dyadic predicate of the form $q(c, d)$, let T_2 be a template for q, let $\tau \in [0, 1]$ be the* **neuro-matching threshold**, *and let f be an embedding function, The* **neuro-matching relation**, *denoted $\simeq$, is defined as follows*

$$\alpha \simeq \beta \textit{ iff } (\beta, x) \in \mathsf{Sim}(\alpha) \textit{ and } \forall (\beta', y) \in \mathsf{Sim}(\alpha), x \geq y$$

where $\mathsf{Sim}(\alpha) = \{(\beta', x) \mid \beta' \in \mathsf{Atoms}(\phi)$ *and* $x > \tau$ *and* $\mathsf{similarity}(f(\mathsf{Inst}(\alpha, T_1)), f(\mathsf{Inst}(\beta', T_2))) = x\}$.

This definition finds the best match in the premise (if there is one that has a similarity greater than the neuro-matching threshold) for each atom in the claim: So $\alpha \simeq \beta$ holds when for all the atoms that occur in the premises $\beta_1, \ldots, \beta_n$, and their similarity scores $s_1, \ldots, s_n$, then β is the atom β_i for which $s_i = \max(s_1, \ldots, s_n)$.

Example 3. *Let the text for the premise be "A tiger is walking around a cage.", and the text for the claim be "A tiger is pacing around a cage.", we have the following AMR formulas.*

$$\mathtt{arg0}(\mathtt{walk}, \mathtt{tiger}) \land \mathtt{arg2}(\mathtt{walk}, \mathtt{around}) \\ \land \mathtt{op1}(\mathtt{around}, \mathtt{cage})$$

$$\mathtt{arg0}(\mathtt{pace}, \mathtt{tiger}) \land \mathtt{arg2}(\mathtt{pace}, \mathtt{around}) \\ \land \mathtt{op1}(\mathtt{around}, \mathtt{cage})$$

When $\tau = 0.6$, $\mathtt{arg0}(\mathtt{walk}, \mathtt{tiger}) \simeq \mathtt{arg0}(\mathtt{pace}, \mathtt{tiger})$, *and* $\mathtt{arg2}(\mathtt{walk}, \mathtt{around}) \simeq \mathtt{arg2}(\mathtt{pace}, \mathtt{around})$, *assuming the following similarities given the templates T and T' for* $\mathtt{arg0}$ *and* $\mathtt{arg2}$ *respectively.*

$$\mathsf{similarity}(f(\mathsf{Inst}(\mathtt{arg0}(\mathtt{walk}, \mathtt{tiger}), T)), \\ f(\mathsf{Inst}(\mathtt{arg0}(\mathtt{pace}, \mathtt{tiger})), T))) = 0.8483$$

$$\mathsf{similarity}(f(\mathsf{Inst}(\mathtt{arg2}(\mathtt{walk}, \mathtt{around}), T')), \\ f(\mathsf{Inst}(\mathtt{arg2}(\mathtt{pace}, \mathtt{around})), T'))) = 0.762$$

In general, the $\simeq$ relation is reflexive but not symmetric.

C. Translating AMR formulas into abstract formulas

We now consider how we can translate each AMR formula into an abstract formula. The first aim is to represent each AMR atom by an abstract atom of the form $\mathtt{x_i}$ in order to facilitate representation by a PySAT solver, and the second aim is to take advantage of the neuro-matching relation to simplify the abstract formulae. For example, if we have a premise $\mathtt{x}_1 \land \mathtt{x}_2 \land \mathtt{x}_3$ and a claim $\mathtt{x}_1 \land \mathtt{x}_4$ and we can show that $\mathtt{x}_3$ and $\mathtt{x}_4$ are very similar concepts, then we can change the claim to $\mathtt{x}_1 \land \mathtt{x}_3$, and then show entailment holds using the CNF versions of these relaxed formulas with PySAT.

Given an AMR formula, or an abstract formula, denoted ϕ, let $\mathsf{Atoms}(\phi)$ denote the set of atoms used in ϕ.

Definition 6. *Given a set of AMR formulas Φ, and a neuro-matching relation $\simeq$, the function $g : \mathcal{A} \to \mathcal{P}$ is a* **translation** *for Φ and $\simeq$ iff for all $\phi, \phi' \in \Phi$, for all $\alpha \in \mathsf{Atoms}(\phi)$, for all $\beta \in \mathsf{Atoms}(\phi')$, $\alpha \simeq \beta$ iff $g(\alpha) = g(\beta)$. Also for the true atom $\top$, $g(\top) = \top$.*

The above definition ensures that if the same (or similar according to $\simeq$) atom is used in different formulas in Φ, then they are translated to the same atom in the abstract formulas.

Example 4. *Let* $\phi_1 = \mathtt{arg1}(\mathtt{car}, \mathtt{red}) \land \mathtt{arg2}(\mathtt{car}, \mathtt{fast})$ *and* $\phi_2 = \mathtt{arg1}(\mathtt{car}, \mathtt{red})$. *Let $\Phi = \{\phi_1, \phi_2\}$. A translation g for Φ is* $g(\mathtt{arg1}(\mathtt{car}, \mathtt{red})) = \mathtt{x}_1$ *and* $g(\mathtt{arg2}(\mathtt{car}, \mathtt{fast})) = \mathtt{x}_2$

Next we specify how an AMR formula can be translated by treating the abstract formulas as an abstraction.

Definition 7. *Let Φ be a set of AMR formula and let g be a translation for Φ. For $\phi \in \Phi$, an* **abstraction** *of ϕ is $\mathsf{Abstract}_g(\phi)$ where $\mathsf{Abstract}_g$ is defined as follows: (1) $\mathsf{Abstract}_g(\alpha \land \beta) = \mathsf{Abstract}_g(\alpha) \land \mathsf{Abstract}_g(\beta)$; (2) $\mathsf{Abstract}_g(\neg\alpha) = \neg\mathsf{Abstract}_g(\alpha)$; And (3) $\mathsf{Abstract}_g(\alpha) = g(\alpha)$ when $\alpha \in \mathcal{A}$. For a set Φ, $\mathsf{Abstract}_g(\Phi) = \{\mathsf{Abstract}_g(\phi) \mid \phi \in \Phi\}$.*

Example 5. *Continuing Example 4, an abstraction of ϕ_1 is $\mathtt{x}_1 \land \mathtt{x}_2$ and ϕ_2 is $\mathtt{x}_1$.*

Example 6. *Continuing Example 3,* $g(\mathtt{arg0}(\mathtt{walk}, \mathtt{tiger})) = g(\mathtt{arg0}(\mathtt{pace}, \mathtt{tiger}) = \mathtt{x}_1$, *and* $g(\mathtt{arg2}(\mathtt{walk}, \mathtt{around})) = g(\mathtt{arg2}(\mathtt{pace}, \mathtt{around})) = \mathtt{x}_2$.

So our relaxed propositional entailment method rewrites the formulas that are obtained from the AMR-to-propositional-logic translator, into abstract formulas. These abstract formulas are then rewritten into CNF, and tested for entailment using a PySAT solver (as explained in Section II-B).

D. Contradiction methods

To show whether a premise contradicts a claim, we use the relaxed propositional entailment described above. We also require a notion of forgetting which we motivate next.

Some premises express constraints that are in the form of the negation of a conjunction of atoms. For example, the premise "No animal is active around the tree" might be represented by $\neg(\texttt{arg0}(\texttt{activity}, \texttt{animal}) \wedge \texttt{location}(\texttt{activity}, \texttt{around}) \wedge \texttt{op1}(\texttt{around}, \texttt{tree}))$. Suppose we have the claim "A big cat is moving," which is represented by $\texttt{arg0}(\texttt{move}, \texttt{big cat})$. When we try to prove that the premise contradicts the claim, assuming $\texttt{arg0}(\texttt{activity}, \texttt{animal}) \simeq \texttt{arg0}(\texttt{move}, \texttt{big cat})$ using relaxed propositional entailment, the additional atoms $\texttt{location}(\texttt{activity}, \texttt{around})$ and $\texttt{op1}(\texttt{around}, \texttt{tree})$ appear irrelevant. This irrelevancy means we are unable to show that the premise is inconsistent with the claim. To address this, we introduce a method to forget irrelevant atoms.

Example 7. *Consider the premise* $\neg(x_1 \wedge x_2 \wedge x_3)$ *and claim* $\mathtt{x}_1 \wedge \mathtt{x}_2$ *as abstract formulas. The conjunction* $\mathtt{x}_1 \wedge \mathtt{x}_2 \wedge \neg(\mathtt{x}_1 \wedge \mathtt{x}_2 \wedge \mathtt{x}_3)$ *is consistent. We may regard atom* $\mathtt{x}_3$ *as irrelevant because it is not in the claim. If we forget* $\mathtt{x}_3$*, we have* $\mathtt{x}_1 \wedge \mathtt{x}_2 \wedge \neg(\mathtt{x}_1 \wedge \mathtt{x}_2)$ *which is inconsistent.*

The forgetting method ignores the atoms in the premises that are not in the claim as defined next.

Definition 8. *For premise* ϕ *and claim* ψ*, the* **forgettable atoms** *of* ϕ *and* ψ *are* $\mathsf{Atoms}(\phi) \setminus \mathsf{Atoms}(\psi)$.

Example 8. *Suppose* $\mathsf{Atoms}(\phi) = \{\mathtt{x}_1, \mathtt{x}_2, \mathtt{x}_3, \mathtt{x}_4\}$ *and* $\mathsf{Atoms}(\psi) = \{\mathtt{x}_1, \mathtt{x}_2\}$*. The forgettable atoms of* ϕ *and* ψ *are* $\{\mathtt{x}_1, \mathtt{x}_2, \mathtt{x}_3, \mathtt{x}_4\} \setminus \{\mathtt{x}_1, \mathtt{x}_2\} = \{\mathtt{x}_3, \mathtt{x}_4\}$.

Definition 9. *For a forgettable atom* α*, the* **forget function**, Forget_α*, is defined as follows, where* ϕ *and* ψ *are abstract formulas: (1)* $\mathsf{Forget}_\alpha(\phi \wedge \psi) = \mathsf{Forget}_\alpha(\phi) \wedge \mathsf{Forget}_\alpha(\psi)$*; (2)* $\mathsf{Forget}_\alpha(\neg\beta) = \neg\mathsf{Forget}_\alpha(\beta)$*; (3)* $\mathsf{Forget}_\alpha(\alpha) = \top$*; and (4)* $\mathsf{Forget}_\alpha(\beta) = \beta$ *if* β *is an atom and* $\beta \neq \alpha$*. The forget function is generalized to a set of forgettable atoms* $\Gamma = \{\alpha_1, \ldots, \alpha_n\}$ *as follows:* $\mathsf{Forget}_\Gamma = \mathsf{Forget}_{\alpha_1}(\ldots\ldots\mathsf{Forget}_{\alpha_n}(\phi)..)$.

Example 9. *Suppose* $\alpha = \mathtt{x}_1, \phi = \mathtt{x}_1 \wedge \mathtt{x}_2 \wedge \mathtt{x}_3, \psi = \mathtt{x}_4 \wedge \mathtt{x}_5, \beta = \mathtt{x}_6$*,* $\mathsf{Forget}_\alpha(\phi \wedge \psi) = \mathsf{Forget}_\alpha(\phi) \wedge \mathsf{Forget}_\alpha(\psi) = \mathsf{Forget}_{\mathtt{x}_1}(\mathtt{x}_1 \wedge \mathtt{x}_2 \wedge \mathtt{x}_3) \wedge \mathsf{Forget}_{\mathtt{x}_1}(\mathtt{x}_4 \wedge \mathtt{x}_5) = \mathtt{x}_2 \wedge \mathtt{x}_3 \wedge \mathtt{x}_4 \wedge \mathtt{x}_5$*;* $\mathsf{Forget}_\alpha(\neg\beta) = \neg\mathsf{Forget}_\alpha(\beta) = \neg\mathsf{Forget}_{\mathtt{x}_1}(\mathtt{x}_6) = \neg\mathtt{x}_6$

Example 10. *Assuming the text for the premise is "No tiger is walking around a cage quickly.", and the text for the claim is "A tiger is pacing around a cage.", we have the following AMR formulas.*

$$\neg(\texttt{arg0}(\texttt{walk}, \texttt{tiger}) \wedge \texttt{arg2}(\texttt{walk}, \texttt{around}) \\ \wedge \texttt{op1}(\texttt{around}, \texttt{cage}) \wedge \texttt{mod}(\texttt{quick}, \texttt{walk}))$$

$$\texttt{arg0}(\texttt{pace}, \texttt{tiger}) \wedge \texttt{arg2}(\texttt{pace}, \texttt{around}) \\ \wedge \texttt{op1}(\texttt{around}, \texttt{cage})$$

For the translation function $g(\texttt{arg0}(\texttt{walk}, \texttt{tiger})) = \mathtt{x}_1$, $g(\texttt{arg2}(\texttt{walk}, \texttt{around})) = \mathtt{x}_2$, $g(\texttt{op1}(\texttt{around}, \texttt{cage})) = \mathtt{x}_3$, $g(\texttt{arg0}(\texttt{pace}, \texttt{tiger})) = \mathtt{x}_4$, $g(\texttt{arg2}(\texttt{pace}, \texttt{around})) = \mathtt{x}_5$, $g(\texttt{mod}(\texttt{quick}, \texttt{walk})) = \mathtt{x}_6$*, we obtain.*

$$\mathsf{Abstract}_g(\Phi) = \{\neg(\mathtt{x}_1 \wedge \mathtt{x}_2 \wedge \mathtt{x}_3 \wedge \mathtt{x}_6)\}$$
$$\mathsf{Abstract}_g(\alpha) = \mathtt{x}_3 \wedge \mathtt{x}_4 \wedge \mathtt{x}_5$$

From Example 6, we have $g(\texttt{arg0}(\texttt{walk}, \texttt{tiger})) = g(\texttt{arg0}(\texttt{pace}, \texttt{tiger}) = \mathtt{x}_1$ *and* $g(\texttt{arg2}(\texttt{walk}, \texttt{around})) = g(\texttt{arg2}(\texttt{pace}, \texttt{around})) = \mathtt{x}_2$*. Therefore we have*

$$\mathsf{Abstract}_g(\Phi) = \{\neg(\mathtt{x}_1 \wedge \mathtt{x}_2 \wedge \mathtt{x}_3 \wedge \mathtt{x}_6)\}$$
$$\mathsf{Abstract}_g(\alpha) = \mathtt{x}_1 \wedge \mathtt{x}_2 \wedge \mathtt{x}_3$$

Then according to Definition 8, the The forgettable atoms of $\mathsf{Atoms}(\Phi) \setminus \mathsf{Atoms}(\alpha)$ *is* $\{\mathtt{x}_1, \mathtt{x}_2, \mathtt{x}_3, \mathtt{x}_6\} \setminus \{\mathtt{x}_1, \mathtt{x}_2, \mathtt{x}_3\} = \{\mathtt{x}_6\}$*. Therefore we can prove this pair of premise and claim is contradictory as* $\mathtt{x}_1 \wedge \mathtt{x}_2 \wedge \mathtt{x}_3 \neg(\mathtt{x}_1 \wedge \mathtt{x}_2 \wedge \mathtt{x}_3)$ *is inconsistent.*

The forgetting method allows more pairs of premise and claim to be classed as inconsistent. In our empirical study, we investigate the trade-off of recall and precision for this.

Predicted class	$\phi \wedge \neg\psi$	$\phi^* \wedge \psi$
Entailment	False	True
Contradiction	True	False
Neutral	True	True

TABLE II: Classification of relations. Let ϕ (resp. ψ) be the abstract formula for a premise (resp. claim), and let ϕ^* be $\mathsf{Forget}_\Gamma(\phi)$ where Γ is the set of forgettable atoms of ϕ and ψ. For columns 2 and 3, PySAT returns True if the formula is consistent, and returns False if it is inconsistent. So for instance the entailment relation holds when $\phi \wedge \neg\psi$ is False and $\phi^* \wedge \psi$ is True.

E. Classification of relations

The aim of our pipeline is to identify the relationship between a premise ϕ and a claim ψ. We translate our AMR formulas into abstract formula using the neuro-matching relation (as explained in Sections III-C) and using forgetting (as explained in III-D) to give the relaxed formulas which we use in automated reasoning (as explained in Section II-B) to check entailment and contradiction simultaneously, and then classify according to Table II, where Γ is the set of forgettable atoms of formulas ϕ and ψ, and ϕ^* denotes $\mathsf{Forget}_\Gamma(\phi)$.

There is another possibility for classification of relations (in Table II), which is when $\phi \wedge \neg\psi$ is inconsistent (False) and $\phi^* \wedge \psi$ is inconsistent (False). This would arise if ϕ is inconsistent. We investigated this by randomly selecting 1000 data items from the both dataset, and in all cases, ϕ was consistent. This suggests inconsistent premises ϕ are a rare or potentially non-existent occurrence in the datasets used for our studies. We may also get both $\phi \wedge \neg\psi$ and $\phi^* \wedge \psi$ being False due to the forgetting method. Consider premise $\phi = \mathtt{x}_1 \wedge \neg\mathtt{x}_2$ and claim $\psi = \mathtt{x}_1$. So $\phi \wedge \neg\psi$ is inconsistent (False). Because $\mathtt{x}_2$ is a forgettable atom, we have $\phi^* = \mathtt{x}_1 \wedge \neg\mathtt{x}_2 = \mathtt{x}_1 \wedge \neg\top = \mathtt{x}_1 \wedge \bot$, so $\phi^* \wedge \psi = \mathtt{x}_1 \wedge \bot \wedge \mathtt{x}_1$ is also inconsistent (False). However,

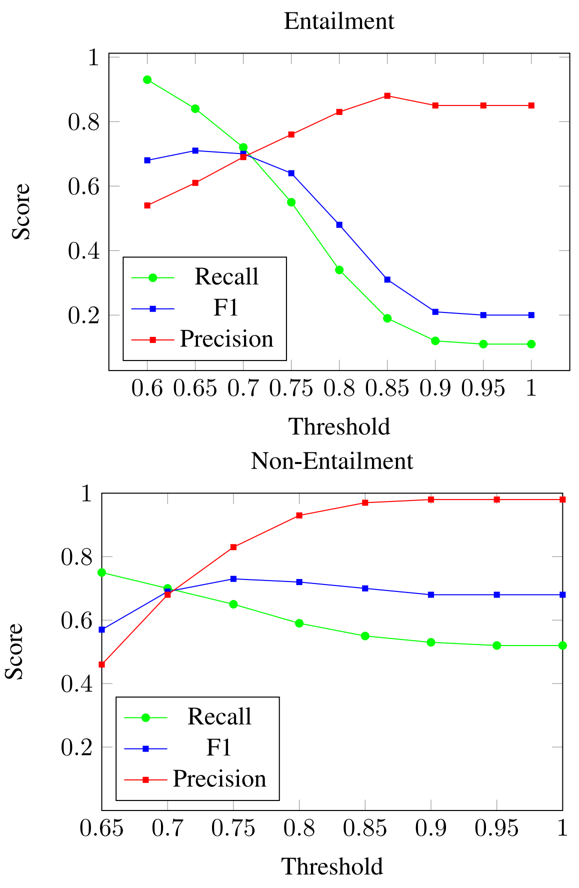

Fig. 3: SICK dataset: Recall, precision, and F1 score, versus neuro-matching threshold.

in the datasets, we found no instances where both $\phi \wedge \neg\psi$ and $\phi^* \wedge \psi$ are inconsistent (False).

IV. Datasets

We used two datasets (SICK and STS-B) for our evaluations. The **SICK** (Sentences Involving Compositional Knowledge) dataset is for compositional distributional semantics covering diverse lexical, syntactic and semantic phenomena [21]. It comprises 9840 items of data. where each item consists of a premise, a claim, and a label (entailment, contradiction, or neutral). The **STS-B** (Semantic Textual Similarity Benchmark) dataset is a resource designed to evaluate a models' ability to assess semantic similarity between pairs of text snippets or sentences [22]. It comprises 8628 items of data where each item consists of two sentences, and a data similarity score normalized to a value in the $[0, 1]$ interval.

V. Result

For the STS-B dataset, we adopt the ordinal annotation scale from [23], where a data similarity score of 0 indicates *no meaning overlap* and a data similarity score of 1 indicates *meaning equivalence*. Furthermore, the annotation guidelines specify that a data similarity score of 0.8 means *the two sentences are mostly equivalent, but some unimportant details*

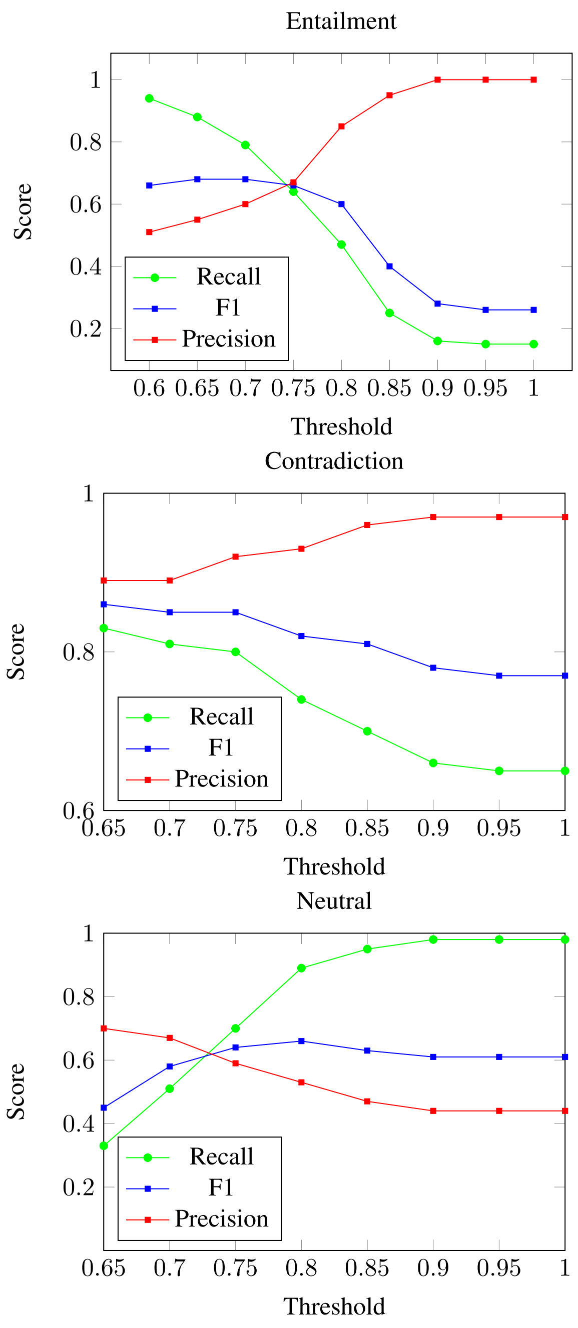

Fig. 4: SICK dataset: Recall, precision, and F1 score, versus neuro-matching threshold.

differ and a data similarity score of 1 means *the two sentences are completely equivalent, as they mean the same thing*. In our task formulation, we introduce the notion of a **dataset threshold** such that if the data similarity score is greater than or equal to the dataset threshold, then we treat the item (i.e. the pair of sentences) in the dataset as **entailment** and otherwise as **non-entailment**. We use this binary data in our neuro-symbolic pipeline, so that **Entailment** means the prediction is Entailment and **Non-entailment** means the prediction is either contradiction and neutral.

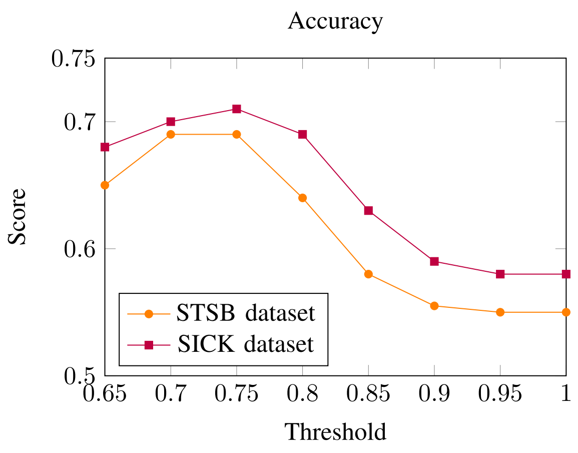

Fig. 5: Overall accuracy of the two datasets.

We investigated how performance changes with different neuro-matching thresholds τ. We assume a maximum sentence length of 20 words and a dataset threshold of 0.8. We tested seven neuro-matching thresholds τ from 0.6 to 1, and randomly selected 1000 data items with two labels (500 each) from the STSB dataset. In Figure 3, the recall of entailment decreases significantly as τ increases from 0.6 to 0.9, while the precision of entailment rises gradually from $\tau = 0.6$ to $\tau = 0.85$ before decreasing slightly. The recall of non-entailment decreases slightly as τ changes from 0.6 to 0.85, whereas the precision of contradiction increases significantly until $\tau = 0.85$. When τ increases from 0.9 to 1, the pipeline has significantly reduced relaxation, leading to minimal performance changes for both classes. This experiment suggests that the relaxation methods are effective for improving entailment and non-entailment identification in the STSB dataset.

For the SICK dataset, we assumed a maximum sentence length of 20 words and tested seven neuro-matching thresholds τ (0.6–1) on 1500 randomly selected data items from data, distributed across three labels (500 each). In Figure 4, the recall of entailment shows a similar trend to the STSB dataset. The recall of contradiction decreases slightly as τ increases from 0.6 to 0.75, then declines significantly until τ reaches 0.9. Conversely, the precision of contradiction increases gradually until $\tau = 0.85$. The recall of the neutral class rises significantly as τ changes from 0.6 to 0.8, followed by a slight increase until $\tau = 0.9$, while its precision improves from 0.62 to 0.7 as τ increases from 0.6 to 0.65 before decreasing slightly as τ approaches 0.9. Similar to the STSB dataset, increasing τ from 0.9 to 1 results in minimal changes for both classes.

Figure 5 shows the overall accuracy (encompassing all classes) versus the neuro-matching threshold. Here the optimal neuro-matching threshold is determined to be $\tau = 0.7$ for the STSB dataset and $\tau = 0.75$ for the SICK dataset.

VI. Structured Argumentation graphs

We now consider how we can apply our neuro-symbolic pipeline to construct a graphical representation of simple natural language arguments. We define a **structured argument graph** as a directed graph where each node is a premise or claim, and each arc denotes a logical relationship between a pair of nodes. For each node, we label it as a premise or claim, and give the original sentence (from the dataset or extracted using argument mining methods), with the AMR formula of that sentence. For each arc, we label it according to Table III. and give the neuro-matching relations and forgotten atoms used in showing the logical relationships between the node as identified by our relaxation methods.

In the structured argument graph, when we can see the relaxations that have been assumed (neuro-matching relations and forgotten atoms), the user can then judge whether the relaxations are reasonable from a commonsense perspective. Moreover, the structured argument graph is useful to determining the underlying argumentative structure of a sequence of sentences. In future work, we will investigate the use of syntactic sugar to improve the presentation of the logical formulas. As we have only demonstrated how our neuro-symbolic pipeline works with simple natural language arguments, for our next step, we want to consider more complex natural language arguments involving multiple sentences.

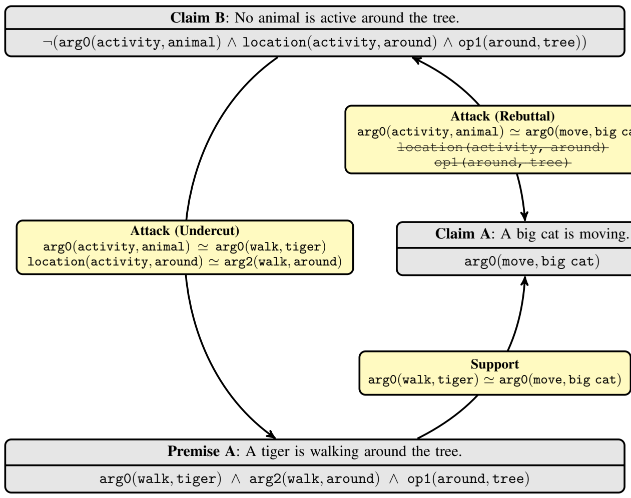
Claim B: No animal is active around the tree.
¬(arg0(activity, animal) ∧ location(activity, around) ∧ op1(around, tree))
Attack (Rebuttal)
arg0(activity, animal) ≃ arg0(move, big cat)
~~location(activity, around)~~
~~op1(around, tree)~~
Attack (Undercut)
arg0(activity, animal) ≃ arg0(walk, tiger)
location(activity, around) ≃ arg2(walk, around)
Claim A: A big cat is moving.
arg0(move, big cat)
Support
arg0(walk, tiger) ≃ arg0(move, big cat)
Premise A: A tiger is walking around the tree.
arg0(walk, tiger) ∧ arg2(walk, around) ∧ op1(around, tree)

Fig. 6: A structured argument graph for a discussion where the first agent says *A big cat is moving*, with justification that *A tiger is walking around the tree*, and then the second agent says *No animal is active around the tree*. For each arc, the neuro-matching relation, and the forgotten atoms (crossed out), that are used for showing the relationship, are in the yellow label.

Source node	Target node	Relationship	Arc label
Premise	Claim	Entailment	Support
Premise	Claim	Neutral	Neutral
Claim	Claim	Contradiction	Rebuttal
Claim	Premise	Contradiction	Undercut

TABLE III: Argumentative labelling of arcs in structured argument graphs according to the type of node for source and target, and the logical relationship according to our pipeline.

VII. Discussion

This study presents a neuro-symbolic pipeline that integrates a pre-trained Abstract Meaning Representation (AMR) parser, with a relaxation method based on word embeddings, and a SAT solver, to detect and explain entailment and contradiction relationships between premises and claims. Evaluated on two simple datasets, the approach achieves a balanced precision-recall performance, addressing some of the limitations of a purely symbolic reasoning in argument analysis. Even though the sentences are relatively simple, our proposal offers the first scalable solution to translating natural language arguments into propositional logic arguments.

The evaluation datasets (STS-B and SICK) were developed for the NLP text entailment task. Symbolic methods for that task include *logical inference-based methods* [24], [25], *lexical-based methods* [26], [27], and *syntax-based methods* [28]. In the past, the primary disadvantage of symbolic methods for text entailment is their lack of scalability and reliance on additional information to handle sentences involving common-sense knowledge. More recently, methods applied to these datasets involve training *machine learning* (ML) [29] or *deep learning* (DL) [30], [31] models. However, these ML/DL models operate as black boxes, and do not output logical formula which we require for argumentation. Because our neuro-symbolic pipeline has the advantage of LLMs for translating sentences into AMR, and the advantage of using LLM word embeddings for identifying similarity between atoms, we have a better compromise than the above methods for our task of generating structured argument graphs from simple natural language arguments.

Future work will extend our neuro-symbolic pipeline to more complex premises and claims derived from argument mining [4], including multiple sentences. It will also include exploring alternatives to AMR for meaning representation (see [32] for a review) that may for instance support the representation of conditional, causal, or temporal, statements, or beliefs and attitudes, which are all important notions in natural language argumentation, and may call for a modal logic formalization (e.g. [33]).

References

[1] K. Atkinson *et al.*, "Towards artificial argumentation," *AI Magazine*, vol. 38, pp. 25–36, 2017.

[2] J. Lawrence and C. Reed, "Argument mining: A survey," *Computational Linguistics*, vol. 45, pp. 765–818, Dec. 2019.

[3] C. Stab and I. Gurevych, "Identifying argumentative discourse structures in persuasive essays," in *Proceedings of the EMNLP*, 2014, pp. 46–56.

[4] R. Mochales and M.-F. Moens, "Argumentation mining," *Artificial Intelligence and Law*, vol. 19, p. 1–22, 2011.

[5] M. Lippi and P. Torroni, "Argumentation mining: State of the art and emerging trends," *ACM Trans. Internet Technol.*, vol. 16, mar 2016.

[6] P. Besnard *et al.*, "Introduction to structured argumentation," *Argument & Computation*, vol. 5, pp. 1–4, 2014.

[7] P. M. Dung, R. A. Kowalski, and F. Toni, "Assumption-based argumentation," in *Argumentation in Artificial Intelligence*, 2009, pp. 199–218.

[8] S. Modgil and H. Prakken, "The ASPIC+ framework for structured argumentation: a tutorial," *Argument & Computation*, vol. 5, pp. 31–62, 2014.

[9] P. Besnard and A. Hunter, "A Review of Argumentation Based on Deductive Arguments," in *Handbook of Formal Argumentation*. College Publications, 2018, pp. 437–484.

[10] I. Langkilde and K. Knight, "Generation that exploits corpus-based statistical knowledge," in *Proceedings of ACL*, Aug. 1998, pp. 704–710.

[11] R. T. Kasper, "A flexible interface for linking applications to penman's sentence generator," in *Proceedings of the Workshop on Speech and Natural Language*, ser. HLT '89. USA: Association for Computational Linguistics, 1989, p. 153–158.

[12] L. Banarescu *et al.*, "Abstract Meaning Representation for sembanking," in *Proceedings of Linguistic Annotation Workshop and Interoperability with Discourse*, 2013, pp. 178–186.

[13] E. Hovy, M. Marcus, M. Palmer, L. Ramshaw, and R. Weischedel, "OntoNotes: The 90% solution," in *Proceedings of NAACL-HLT*, 2006, pp. 57–60.

[14] P. Kingsbury and M. Palmer, "From TreeBank to PropBank," in *Proceedings of LREC*, 2002.

[15] Y. Lee, R. F. Astudillo, T. L. Hoang, T. Naseem, R. Florian, and S. Roukos, "Maximum bayes smatch ensemble distillation for AMR parsing," *CoRR*, vol. abs/2112.07790, 2021.

[16] J. Bos, "Squib: Expressive power of Abstract Meaning Representations," *Computational Linguistics*, vol. 42, pp. 527–535, 2016.

[17] D. Chanin and A. Hunter, "Neuro-symbolic commonsense social reasoning," *arXiv preprint arXiv:2303.08264*, 2023.

[18] A. Meurer *et al.*, "SymPy: symbolic computing in python," *PeerJ Computer Science*, vol. 3, p. e103, 2017.

[19] A. Ignatiev, A. Morgado, and J. Marques-Silva, "PySAT: A Python toolkit for prototyping with SAT oracles," in *SAT*, 2018, pp. 428–437.

[20] S. Xiao, Z. Liu, P. Zhang, N. Muennighoff, D. Lian, and J.-Y. Nie, "C-pack: Packaged resources to advance general chinese embedding," in *Proceedings of SIGIR*, 2024, pp. 641–649.

[21] M. Marelli, L. Bentivogli, M. Baroni, R. Bernardi, S. Menini, and R. Zamparelli, "SemEval-2014: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment," in *Proceedings of SemEval*, 2014, pp. 1–8.

[22] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia, "SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation," in *Proceedings of SemEval)*, 2017, pp. 1–14.

[23] E. Agirre, D. Cer, M. Diab, A. Gonzalez-Agirre, and W. Guo, "*SEM 2013 shared task: Semantic textual similarity," in *Proceedings of Joint Conference on Lexical and Computational Semantics*, 2013, pp. 32–43.

[24] J. Bos and K. Markert, "Recognising textual entailment with logical inference," in *Proceedings of EMNLP*, 2005, p. 628–635.

[25] A. Wotzlaw and R. Coote, "A logic-based approach for recognizing textual entailment supported by ontological background knowledge," *CoRR*, vol. abs/1310.4938, 2013.

[26] O. Glickman, I. Dagan, and M. Koppel, "A lexical alignment model for probabilistic textual entailment," in *Proceedings of the First International Conference on Machine Learning Challenges*, ser. LNCS, vol. 3944, 2005, p. 287–298.

[27] V. Jijkoun, M. de Rijke *et al.*, "Recognizing textual entailment using lexical similarity," in *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*. Citeseer, 2005, pp. 73–76.

[28] M. Alabbas, "Textual entailment for modern standard arabic," Ph.D. dissertation, University of Manchester, 2013.

[29] M. Ben-sghaier, W. Bakari, and M. Neji, "Arabic logic textual entailment with feature extraction and combination," in *Intelligent Systems Design and Applications*, 2020, pp. 400–409.

[30] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A large annotated corpus for learning natural language inference," *CoRR*, vol. abs/1508.05326, 2015.

[31] T. Rocktaschel, E. Grefenstette, K. M. Hermann, T. Kocisky, and P. Blunsom, "Reasoning about entailment with neural attention," in *Proceedings of ICLR*, 2016.

[32] Z. Sadeddine, J. Opitz, and F. Suchanek, "A survey of meaning representations – from theory to practical utility," in *Proceedings of NAACL*, 2024, pp. 2877–2892.

[33] C. Leturc and G. Bonnet, "Using n-ary multi-modal logics in argumentation frameworks to reason about ethics," *AI Communications*, vol. 37, pp. 323–355, 2024.