Can Language Models Learn Embeddings of Propositional Logic Assertions?

Nurul Fajrin Ariyani[1], Zied Bouraoui[2], Richard Booth[1],Steven Schockaert[1]
[1]Cardiff NLP, Cardiff University, UK, [2]CRIL-CNRS & University of Artois, France
{ariyaninf, boothr2, schockaerts1}@cardiff.ac.uk, zied.bouraoui@cril.fr

Abstract

Natural language offers an appealing alternative to formal logics as a vehicle for representing knowledge. However, using natural language means that standard methods for automated reasoning can no longer be used. A popular solution is to use transformer-based language models (LMs) to directly reason about knowledge expressed in natural language, but this has two important limitations. First, the set of premises is often too large to be directly processed by the LM. This means that we need a retrieval strategy which can select the most relevant premises when trying to infer some conclusion. Second, LMs have been found to learn shortcuts and thus lack robustness, putting in doubt to what extent they actually understand the knowledge that is expressed. Given these limitations, we explore the following alternative: rather than using LMs to perform reasoning directly, we use them to learn embeddings of individual assertions. Reasoning is then carried out by manipulating the learned embeddings. We show that this strategy is feasible to some extent while at the same time also highlighting the limitations of directly fine-tuning LMs to learn the required embeddings.

Keywords: formula embeddings, language models, deductive reasoning, propositional logic

1. Introduction

Many Natural Language Processing (NLP) tasks involve some form of reasoning. For instance, answering scientific or technical questions typically involves using available knowledge (e.g. heat travels through a thermal conductor, steel is made of metal, metal is a thermal conductor) to infer specific facts (e.g. heat travels through a steel spoon) (Mihaylov et al., 2018). When such inferences only involve general knowledge, recent Large Language Models (LLMs) can often perform such reasoning steps implicitly. However, many knowledge-intensive tasks require reasoning with application-specific facts and rules. Consider, for instance, the problem of checking whether a given scenario or process is compliant with some policy (Zhong et al., 2018; Bonatti et al., 2020). Current approaches for addressing such problems mostly rely on formal knowledge representation techniques. However, formalising policies is non-trivial and highly time-consuming, severely limiting the real-world impact of AI techniques in this area. Ideally, we would like to directly reason about the original, natural language specification of the policy.

Using Language Models (LMs) in such applications is challenging. First, we cannot simply feed the available knowledge as input to an LLM, because the set of assertions is simply too large (e.g. real-world policies are often hundreds of pages long). Thus, we need a kind of retrieval mechanism to select the knowledge that is needed to complete a given reasoning task. Second, the ability of language models to faithfully carry out complex reasoning tasks remains limited (Creswell et al., 2023). This problem is exacerbated when only smaller LMs can be used, which is often the case in practice due to the prohibitive cost of using LLMs. While smaller BERT-style (Devlin et al., 2019) LMs can be fine-tuned to carry out reasoning tasks (Clark et al., 2020), such models tend to learn statistical artefacts of the training set rather than the meaning of logical formulas (Zhang et al., 2023).

To address these concerns, in this paper, we study the possibility of learning embeddings that capture the meaning of individual assertions. In line with recent work (Clark et al., 2020; Zhang et al., 2023), we focus on assertions that can be formalised using propositional logic and which are expressed in a simplified form of natural language. We focus on this simplified setting as it allows us to analyse the limitations (and potential) of LMs for this task more clearly. In contrast to this previous work, we focus on embedding individual formulas, rather than using a joint encoder where all premises and the hypothesis are fed to the model together.

Formally, we want to learn an encoder $\textsc{Enc}$ which allows us to check whether some premises $p_1, ..., p_k$ entail a hypothesis h by first pooling the embeddings $\textsc{Enc}(p_1), ..., \textsc{Enc}(p_k)$ in some way and then comparing the resulting embedding with $\textsc{Enc}(h)$. While this approach may yield slightly lower performance than a joint encoder, it offers significant advantages. First, the encoder $\textsc{Enc}$ can straightforwardly be used to develop a bi-encoder retrieval model, allowing us to efficiently retrieve relevant pre-computed premises from a potentially large knowledge base. Second, we hypothesise

LREC-COLING 2024, pages 2766–2776
20-25 May, 2024. © 2024 ELRA Language Resource Association: CC BY-NC 4.0

that this model will be less prone to shortcut learning: since each formula is embedded separately, the model is intuitively forced to capture the meaning of the formulas.

2. Related Work

Reasoning with Language Models. There are several major lines of research that are focused on the reasoning abilities of NLP models. One line focuses on the problem of Natural Language Inference (NLI), which requires models to decide the logical relationship between a premise and a hypothesis, both formulated in natural language (Bos and Markert, 2005; Bowman et al., 2015). The notion of entailment which is used in this setting is inherently informal: the focus is on deciding whether a human who believes the premise would also assume that the hypothesis is likely. A second research line is essentially focused on decomposing questions that are difficult to answer directly. In earlier work on multi-hop question answering (Yang et al., 2018; Welbl et al., 2018), the actual reasoning aspect was often more or less straightforward. In more recent benchmarks, however, it can be harder to infer which reasoning steps are needed (Geva et al., 2021). Finally, there is also a large body of work that focuses on more specialised reasoning tasks, such as numerical reasoning (Roy and Roth, 2015; Cobbe et al., 2021; Amini et al., 2019). While important challenges remain, scaling up LLMs has led to significant improvements in the reasoning abilities of NLP systems, especially with strategies such as chain-of-thought prompting (Wei et al., 2022), where the model is given step-by-step explanations of how similar problem instances can be solved.

Logical Reasoning with Neural Networks. The aforementioned works focus on informal reasoning. The question of whether LMs, or neural networks more generally, are able to learn the semantics of formal logics has also been addressed. For instance, within the context of automated theorem proving, there has been considerable interest in the problem of learning embeddings of logical formulas to help with premise selection (Wang et al., 2017; Paliwal et al., 2020; Crouse et al., 2019). In this setting, formulas are represented in symbolic form (e.g. as a parse tree). Graph Neural Networks (GNN) are typically used for learning the embeddings, although variants of LSTMs have been used as well (Irving et al., 2016; Loos et al., 2017). More recently, the possibility of using transformer architectures for interpreting formal languages has also been considered (Bhattamishra et al., 2020). While our work is also partially motivated in terms of premise selection, we focus on knowledge that is expressed in natural language and we rely on fine-tuned LMs to learn formula embeddings. Beyond learning formula embeddings, several authors have studied the use of neural networks for directly solving formal reasoning tasks, including predicting satisfiability (Selsam et al., 2019), generating satisfying traces of temporal logic formulas (Hahn et al., 2021), and weighted model counting (Abboud et al., 2020).

Logical Reasoning with Language Models. Within NLP, starting with Clark et al. (2020), several authors have studied the ability of fine-tuned LMs to formally reason with natural language encodings of logical formulas. Clark et al. (2020) found that LMs can learn to reason with Horn rules with almost perfect accuracy, where the rules involved were presented in a simplified natural language. However, recent work has suggested that this is mostly due to a form of shortcut learning (Zhang et al., 2023). In particular, when test examples were sampled from a different distribution, performance was found to drop substantially, suggesting that the models do not learn to reason in a systematic way. Richardson and Sabharwal (2022) studied the performance of LMs on hard instances of propositional satisfiability, where clauses were again presented in a simplified natural language. They reported generally strong results but highlighted the need for a careful selection of training data and found that models struggled to generalise to larger problem instances. Finally, to enable more robust inferences, some authors have analysed the use of LMs for generating step-by-step proofs (Saha et al., 2020; Tafjord et al., 2021; Creswell et al., 2023). The underlying idea is that LMs are capable of making accurate one-step inferences, which can be chained by iteratively applying the same model.

3. Representing Formulas as Embeddings

We first discuss the problem of learning formula embeddings from a theoretical point of view, to better understand under what conditions it is possible to do formal reasoning by manipulating such embeddings. In this section, we will assume that formulas are encoded in propositional logic.

Basic Notions. Propositional logic formulas are built from a set of atoms (or atomic propositions) *At* using the connectives $\wedge$, $\vee$, $\neg$ and $\rightarrow$ in the usual way. For instance, $a \wedge b \rightarrow \neg c$ (with $a, b, c \in$ *At*) intuitively means that c is false whenever a and b are true. A *literal* l is either an atom a or a negated atom $\neg a$ (with $a \in$ *At*). In the former case, we say that l is a positive literal; otherwise, we call l a negative literal. A *clause* is a disjunction of literals. A *Horn clause* is a clause that has at most one positive literal. An interpretation ω is a mapping from At to the set $\{$*true*, *false*$\}$, intuitively specifying which of the atoms in *At* is true. Interpretations are extended

to arbitrary formulas in the usual way. We write $\omega \models \alpha$, for an interpretation ω and a propositional formula α, to denote that ω makes the formula α true. In such a case, we say that ω is a model of α. We say that a set of formulas $\{\alpha_1, ..., \alpha_k\}$ entails the formula β, written $\{\alpha_1, ..., \alpha_k\} \models \beta$ if every model of $\alpha_1, ..., \alpha_k$ is also a model of β, i.e. if $\omega \models \alpha_1 \wedge ... \wedge \alpha_k$ implies $\omega \models \beta$. If α and β have the same models, we say that they are *equivalent*. For instance, the Horn clause $\neg a_1 \vee ... \vee \neg a_m \vee b$ is equivalent to the implication $a_1 \wedge ... \wedge a_m \rightarrow b$. Implications of the latter form are called Horn rules.

Formula embeddings. A formula embedding ϕ is a mapping from propositional formulas to $\mathbb{R}^n$. We are interested in learning formula embeddings that support reasoning. Specifically, we want to find a formula embedding ϕ and a scoring function ψ such that $\psi(\phi(\alpha), \phi(\beta)) \geq 0$ iff $\alpha \models \beta$. First, we may wonder if it is actually possible. If the set of atoms *At* is finite, this is indeed the case. For instance, let $\omega_1, ..., \omega_n$ be an enumeration of all interpretations, with $n = 2^{|At|}$ and let us write $(a_1, ..., a_n)$ for the embedding of formula α, defined as:

$$a_i = \begin{cases} 1 & \text{if } \omega_i \models \alpha \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Let $(b_1, ..., b_n) \in \mathbb{R}^n$ be the embedding of a formula β, defined in the same way. Then we have $\alpha \models \beta$ iff $\forall i \,.\, a_i \leq b_i$, which is equivalent to:

$$-\sum_i \text{ReLU}(a_i - b_i) \geq 0 \tag{2}$$

Note that the latter condition holds iff $\sum_i \text{ReLU}(a_i - b_i) = 0$. Furthermore note that with this choice of embedding function, there is a close connection between logical conjunction and min-pooling, i.e. we have:

$$\phi(\alpha_1 \wedge ... \wedge \alpha_k) = \min(\phi(\alpha_1), ..., \phi(\alpha_k))$$

where the minimum is applied component-wise. This is appealing, as it means we can decide $\alpha_1 \wedge ... \wedge \alpha_k \models \beta$ by embedding the premises α_i separately. In scenarios where we have a large set of candidate premises to choose from, this means that all formula embeddings can be pre-computed.

Limitations of formula embeddings. One question is whether it would be possible to check entailment using a linear scoring function, i.e. whether there can be a linear function ψ such that for any formulas α and β we have $\psi(\phi(\alpha) \oplus \phi(\beta)) \geq 0$ iff $\alpha \models \beta$, where we write $\oplus$ for vector concatenation. Unfortunately, this is not possible. In particular, it is easy to verify that the following inequalities cannot be simultaneously satisfied if ψ is linear: $\psi(\phi(\alpha) \oplus \phi(\alpha)) \geq 0$, $\psi(\phi(\beta) \oplus \phi(\beta)) \geq 0$, $\psi(\phi(\alpha) \oplus \phi(\beta)) < 0$, $\psi(\phi(\beta) \oplus \phi(\alpha)) < 0$. We may also wonder whether it is possible to use formula embeddings with fewer than $2^{|At|}$ dimensions. Without further constraints, this is clearly the case. However, as discussed above, it is highly desirable that $\phi(\alpha_1 \wedge ... \wedge \alpha_k) = f(\phi(\alpha_1), ..., \phi(\alpha_k))$ holds, for some standard pooling function f. In that case, unfortunately, it is not possible to use fewer than $2^{|At|}$ dimensions (Schockaert, 2023).

Approximate formula embeddings. Clearly, using embeddings with $2^{|At|}$ dimensions is not usually feasible. Moreover, in practice, the set of atoms is not even fixed a priori, which means that we cannot design an embedding function that will always capture propositional entailment faithfully. However, we can still design embeddings that capture the meaning of logical formulas in an approximate way. In particular, let h be a hashing function, which maps each atom to a value between 1 and ℓ, for some $\ell \in \mathbb{N}$. For each formula α, we can consider a reduced formula α^* in which each occurrence of an atom $a \in$ *At* is replaced by the atom $x_{h(a)}$. Every reduced formula is thus a propositional formula over the set of atoms $At^* = \{x_1, ..., x_\ell\}$. We can then model entailment of these formulas as in (2), using embeddings with 2^ℓ dimensions. Assuming the hashing function assigns each value with the same probability, the chance of two distinct atoms being assigned the same hash is given by $\frac{1}{\ell}$. If α and β together mention m distinct atoms, the probability that these atoms are mapped to distinct hashes is given by $\frac{\ell!}{\ell^m \cdot (\ell - m)!}$. If ℓ is sufficiently large, relative to m, the probability that all atoms are mapped to distinct hashes is thus close to 1. In such a case, $\alpha^* \models \beta^*$ is equivalent to $\alpha \models \beta$. We thus expect reasoning with formula embeddings to be accurate as long as the total number of atoms that appear in the formulas involved is sufficiently small, regardless of the overall size of *At*.

4. Encoding Strategies

We now describe the models we rely on for learning formula embeddings. The input to these models consists of a set of premises $\{p_1, ..., p_k\}$ together with a hypothesis h, all of which are encoded as natural language assertions.

Order Embeddings. Our main model is inspired by the embedding defined in (1), and in particular, by its relationship to the Order Embeddings model from Vendrov et al. (2016). First, each natural language assertion p is encoded as follows:

$$\text{Enc}(p) = \phi(\text{LM}(p))$$

where LM is a fine-tuned LM. Specifically, we use BERT to encode the assertion p and then average the output tokens to get a fixed-dimensional embedding. This embedding is then passed to a

feedforward network ϕ. We check whether α entails β by checking whether each coordinate in the embedding of α is smaller than the corresponding coordinate in the embedding of β. We represent a set of premises P as $\mathsf{Enc}(P) = \min_{i=1}^{k} \mathsf{Enc}(p_i)$, where the minimum is applied component-wise. We write $\mathsf{Enc}_i(P)$ and $\mathsf{Enc}_i(h)$ for the ith coordinate of $\mathsf{Enc}(P)$ and $\mathsf{Enc}(h)$ respectively. We consider the following scoring function:

$$\textit{score}(P, h) = \sum_{i=1}^{n} \max\left(0, \mathsf{Enc}_i(P) - \mathsf{Enc}_i(h)\right)^2$$

If the set of premises P entails the hypothesis h, then we should have $\mathsf{Enc}_i(P) \leq \mathsf{Enc}_i(h)$ for every coordinate $i \in \{1, ..., n\}$. In other words, $\textit{score}(P, h)$ should be 0 if $\{p_1, ..., p_k\}$ entails h, and should be strictly positive otherwise. This leads to the following margin loss (Vendrov et al., 2016):

$$\mathcal{L} = \sum_{(P,h)\in T^+} \textit{score}(\mathsf{Enc}(P), \mathsf{Enc}(h)) + \sum_{(P,h)\in T^-} \max(0, \Delta - \textit{score}(\mathsf{Enc}(P), \mathsf{Enc}(h)))$$

Here T^+ represents a set of positive examples (P, h), where P logically entails h, whereas T^- represents a set of negative examples. At test time, an instance (P, h) is classified as positive if $\textit{score}(\mathsf{Enc}(P), \mathsf{Enc}(h)) \leq \lambda$. In our experiments, we tune $\lambda \in [0, \Delta]$ on the validation set.

Joint Premise Order Embeddings. One key design choice in the Order Embeddings model is the fact that a set of premises P is encoded by min-pooling the embeddings of the individual premises. This has several advantages, but as we pointed out in Section 3, encoding formulas in this way requires high-dimensional embeddings. For this reason, we also experiment with a variant in which the set of premises is jointly encoded. Specifically, we now assume that we have two separate encoders: one for encoding sets of premises and one for encoding the hypothesis:

$$\mathsf{Enc}_{\textit{prem}}(P) = \phi_1(\mathsf{LM}_{\textit{prem}}(P))$$
$$\mathsf{Enc}_{\textit{hyp}}(h) = \phi_2(\mathsf{LM}_{\textit{hyp}}(h))$$

The encoder $\mathsf{Enc}_{\textit{hyp}}$ is used to encode an individual hypothesis, as before. However, to encode the set of premises P we now simply concatenate them, separated by the [SEP] token, and feed the result to another fine-tuned language model $\mathsf{LM}_{\textit{prem}}$. We will consider two variants of this model: one where the parameters of the two encoders are tied, i.e. $\phi_1 = \phi_2$ and $\mathsf{LM}_{\textit{prem}} = \mathsf{LM}_{\textit{hyp}}$, and one where the two encoders use different parameters. The scoring function and loss are then defined as before, i.e. we only change how sets of premises are encoded.

Joint Premise Bi-Encoder. We will also experiment with a variant that uses a standard bi-encoder model. In this case, we train the model using binary cross-entropy, as follows:

$$\mathcal{L} = \sum_{(P,h)\in T^+} \log \sigma(\mathsf{Enc}_{\textit{prem}}(P) \cdot \mathsf{Enc}_{\textit{hyp}}(h)) + \sum_{(P,h)\in T^-} \log(1 - \sigma(\mathsf{Enc}_{\textit{prem}}(P) \cdot \mathsf{Enc}_{\textit{hyp}}(h)))$$

where $\mathsf{Enc}_{\textit{prem}}(P)$ again corresponds to a joint encoding of the set of premises. Note that for this bi-encoder model, we cannot use the same encoder for premises and hypothesis, since this would make the model symmetric, i.e. whenever α entails β we would also have that β entails α.

Joint Encoder. Finally, we will experiment with a model that jointly encodes the premises and hypothesis, which is the approach that has been taken in previous work (Clark et al., 2020; Richardson and Sabharwal, 2022; Zhang et al., 2023). An input with premises $p_1, ..., p_k$ and hypothesis h is then encoded as *[CLS]* p_1. p_2. ... p_k *[SEP]* h *[SEP]*. To predict the entailment label, we feed the output embedding of the [CLS] token to a linear classifier. The model is trained using binary cross-entropy.

5. Datasets

Following previous work (Clark et al., 2020; Richardson and Sabharwal, 2022; Zhang et al., 2023), we will rely on randomly generated training and test sets. Examples are generated by first sampling propositional formulas and then converting these formulas into natural language assertions. As argued by Zhang et al. (2023), when learning to reason from data, it is important that training and test examples are sampled from a different distribution, since models may otherwise perform well by learning statistical artefacts rather than actually solving the intended task. We will thus consider a number of different sampling strategies, as explained in Section 5.1. In Section 5.2, we subsequently describe how we obtain natural language assertions.

5.1. Sampling Propositional Formulas

We consider a number of different sampling strategies. First, following Richardson and Sabharwal (2022), we consider a strategy which is based on sampling random clauses, which we will refer to as **SAT**. We assume that the required number of premises m is specified and that a vocabulary of atoms *At* is also given. We furthermore specify the number ℓ of different atoms from *At* that may occur in the problem instance. Note that this number is known to greatly affect the difficulty of randomly

generated problem instances for traditional reasoning strategies (Mitchell et al., 1992), and might thus also affect how easily LM-based strategies are able to classify the generated problem instances. Moreover, for the embedding-based strategies, the number ℓ is also important because it affects whether learning perfectly faithful embeddings is possible in theory, which is the case when $\ell \leq \lfloor \log_2 d \rfloor$, with d the dimension of the embeddings. Each problem instance consists of m premises and one hypothesis. The premises and hypotheses correspond to clauses of up to 3 literals. To sample a problem instance, we proceed as follows:

- We first sample a subset X of ℓ atoms from *At*.
- We then sample m clauses as follows:
  - We sample the number of literals k in the clause from $\{1, 2, 3\}$.
  - We then randomly choose k atoms from X (with replacement).
  - Finally, each atom is negated with 50% probability.
- We sample the hypothesis in the same way.

To generate a dataset, we also need to determine which problem instances are positive (i.e. in which cases the hypothesis is entailed by the premises). We rely on a standard SAT Solver[1] for this purpose. Depending on the chosen parameters (e.g. the number of clauses and the number of different atoms ℓ), the majority of the generated instances may be positive or negative. To address this, we oversample the minority class, so we always end up with the same number of positive and negative examples in our generated datasets.

Next, we also consider the Label Priority **(LP)** and Rule Priority **(RP)** strategies that were introduced by Zhang et al. (2023). An important difference with the **SAT** strategy is that **LP** and **RP** only sample Horn rules, which makes reasoning typically easier. In the case of **RP**, we first sample a number of facts (i.e. atoms which are said to be true), a number of Horn rules (in the form of clauses of up to 4 literals), and a hypothesis (also in the form of an atom). We then check whether the hypothesis is entailed from the facts and rules to decide the label of the problem instance. In the case of **LP**, we first randomly assign truth values to atoms and then randomly sample facts and Horn rules that are consistent with these truth assignments. Note that due to the use of Horn rules, the datasets sampled using RP and LP do not include any negations. For more details about these strategies, we refer to (Zhang et al., 2023).

[1] https://www.sat4j.org/

Finally, to help us better understand the limitations of the models, we will also experiment with problem instances that correspond to the following standard inference patterns:

Weakening We randomly sample a clause α of $m \in \{2, 3\}$ literals. We obtain the clause β by randomly removing one of the literals in α. We use β as the only premise and α as the hypothesis.

Resolution We randomly sample a clause α of $m \in \{1, 2\}$ literal and a clause β of $m' \in \{1, 2\}$ literals. We also sample an atom $x \in$ *At*. We use $\alpha \vee x$ and $\beta \vee \neg x$ as the two premises and $\alpha \vee \beta$ as the hypothesis.

Contradiction We randomly sample a clause α of $m \in \{1, 2, 3\}$ literals and an atom $x \in$ *At*. We use x and $\neg x$ as the two premises and α as the hypothesis.

Tautology We randomly sample a clause α of $m \in \{1, 2, 3\}$ literals and an atom $x \in$ *At*. We use α as the only premise and $x \vee \neg x$ as the hypothesis.

Modus ponens We sample atoms a_1, a_2 from *At*. We use $\neg a_1 \vee a_2$ and a_1 as the premises and a_2 as the hypothesis.

Note that these problem instances correspond to positive examples. We will also consider three basic types of negative examples:

Strengthening We randomly sample a clause α of $m \in \{2, 3\}$ literals. We obtain the clause β by randomly removing one of the literals in α. We use α as the only premise and β as the hypothesis.

Abduction We sample atoms a_1, a_2 from *At*. We use $\neg a_1 \vee a_2$ and a_2 as the premises and a_1 as the hypothesis.

Random We randomly sample a clause α of $m \in \{1, 2, 3\}$ literals. We also sample a clause β in the same way. We use α as the only premise and β as the hypothesis. Examples where α entails β are discarded since this is aimed to be a strategy for generating negative examples.

5.2. Describing Propositional Formulas

We need a strategy to convert the sampled clauses into natural language. In line with recent work, we use a fixed set of patterns to construct these sentences. This limits the variability in how the formulas are described and thus simplifies the overall problem. We take this approach because our focus is primarily on analysing the reasoning abilities of

Encoding Strategies	SAT	RP	LP
Joint Encoder	**95.6**	**97.2**	88.8
Order Emb. (0L)	91.5	93.5	85.3
Order Emb. (1L, d=768)	92.2	93.1	86.4
Order Emb. (1L, d=1000)	92.8	94.8	**89.0**
Order Emb. (2L, d=1000)	90.5	93.6	84.2
Joint Prem. Order Emb. (tied)	73.0	76.6	53.9
Joint Premise Order Emb.	69.2	69.3	50.0
Joint Premise Bi-encoder	66.7	60.4	50.1

Table 1: Accuracy (%) of different model architectures on the SAT, RP, and LP test sets.

LMs rather than their ability to understand individual assertions.

First, we associate every atom from *At* with a corresponding phrase, such as "Alice is great". We only consider phrases of the form *[person] is [property]*. For negated atoms, we use a phrase of the form *[person] is not [property]* (e.g. "Alice is not great"). To describe a given clause $l_1 \vee l_2 \vee l_3$, we proceed in two steps. First, we rewrite some of the clauses, converting them into logically equivalent formulas involving implications. Specifically, for a clause of the form $l_1 \vee l_2 \vee l_3$ we consider two possible alternatives: $\neg l_1 \rightarrow l_2 \vee l_3$ and $(\neg l_1 \wedge \neg l_2) \rightarrow l_3$. For a clause of the form $l_1 \vee l_2$, we consider $\neg l_1 \rightarrow l_2$ as a possible alternative. We then use fixed templates to describe the resulting formula. For instance, a formula of the form $(\neg l_1 \wedge \neg l_2) \rightarrow l_3$ might be described as "If Alice is not lovable and Alice is not compassionate then Alice is timid".

6. Experimental Analysis

We now experimentally analyse the performance of the proposed strategies[2]. We particularly focus on the following research questions:

- How closely can the performance of the order embeddings model approach that of the common joint encoder?
- Which of the proposed variants performs best? For instance, is there any benefit in jointly embedding all premises, or can we embed premises individually?

Unless specified otherwise, we use the pre-trained BERT$_{\text{base}}$ model[3] to initialise the encoders for all experiments. We use early stop regularization with a learning rate of 2×10^{-5}, a warm-up ratio of 0.067, and a batch size range of [8, 32].

[2]Our code is available at https://github.com/ariyaninf/formula_embeddings.

[3]https://huggingface.co/bert-base-uncased

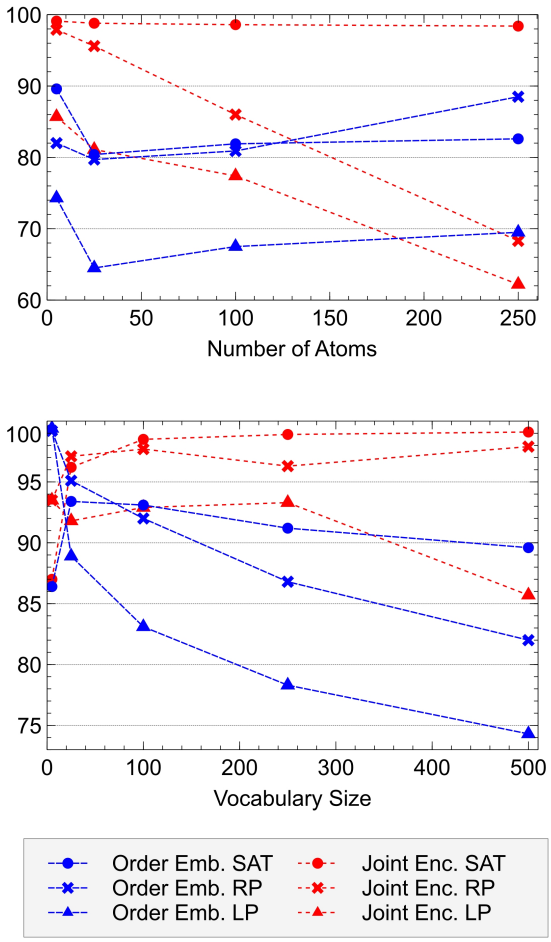

Figure 1: Accuracy (%) for Order Embeddings (1L, d=1000) and Joint Encoder when varying the number of atoms per instance ℓ and the overall vocabulary size $|At|$.

Comparison of Encoding Strategies. First, we compare the effectiveness of the different encoding strategies from Section 4. For this analysis, the models are trained on a set of 100K examples which were sampled using the SAT strategy, with the number of atoms ℓ varying from 5 to 9 and the number of premises m varying from 1 to 10. We use a further 20K SAT examples for validation. Our test sets consist of 10K examples, sampled using SAT, RP or LP. The training, validation and test sets are balanced, meaning that the accuracy of random guessing is 50%. For the order embeddings model, we report results for different variants of the feedforward network ϕ: a variant which omits this network altogether (0L), a variant which uses one layer of d dimensions (1L) and a variant which uses two layers of d dimensions (2L). For the feedforward layers, we use the ReLU activation function. For the other models, no feedforward layers are used.

The results are summarised in Table 1, where we use accuracy as our evaluation metric. As ex-

	Order Embeddings			Joint Encoder		
	SAT	**RP**	**LP**	**SAT**	**RP**	**LP**
SAT	92.8	94.8	88.9	95.6	97.2	88.8
SAT (200K)	**93.3**	96.5	90.1	**96.4**	98.4	94.7
SAT + Easy Examples	91.5	93.9	84.2	96.0	96.4	94.4
SAT + Inference Patterns	84.8	87.3	70.2	95.5	96.1	86.8
Inference Patterns $<$ SAT	88.0	87.0	75.3	95.2	94.7	86.7
RP	55.5	**98.4**	92.5	54.6	**98.6**	96.1
LP	55.7	95.0	**97.4**	53.3	89.3	**99.4**

Table 2: Accuracy (%) for Order Embeddings (1L, d=1000) and Joint Encoder, for different training sets.

		Standard Training						Augmented				Fixed $\lambda = 1$	
		Joint Encoder	**Order Emb.** $(0L)$	**Order Emb.** $(1L, d = 1000)$	**Joint Prem. Order Emb. (tied)**	**Joint Prem. Order Emb.**	**Joint Prem. Bi-Encoder**	**Joint Encoder (SAT + inference)**	**Joint Encoder (inference $<$ SAT)**	**Order Emb. (SAT + inference)**	**Order Emb. (inference $<$ SAT)**	**Order Emb.** $(1L, d = 1000)$	**Order Emb. (inference $<$ SAT)**
Pos	Weakening	96.4	0.0	90.1	42.7	6.6	11.7	99.6	98.7	88.1	89.5	95.9	93.2
	Resolution	93.2	25.4	35.8	37.2	17.3	0.1	99.0	53.1	49.5	39.5	52.4	57.1
	Contradiction	6.1	13.0	89.7	31.0	35.8	64.3	50.5	5.2	83.0	75.5	97.2	90.4
	Tautology	0.9	0.0	1.2	11.3	0.4	3.9	99.5	1.0	82.1	2.7	4.7	4.5
	Modus Ponens	100	15.7	21.0	8.9	57.6	3.3	100	99.2	25.5	14.6	62.2	53.7
Neg	Strengthening	100	100	100	91.5	97.9	100	99.3	99.1	99.5	100	100	100
	Abduction	100	100	99.9	96.3	65.5	95.9	100	100	96.2	99.3	99.5	99.0
	Random	98.5	99.8	98.2	87.0	99.3	96.8	97.8	98.4	94.5	96.6	97.4	96.6

Table 3: Accuracy (%) for test examples based on different inference patterns.

pected, we can see that the order embeddings generally underperform the joint encoder. However, order embeddings achieve the best result on LP. Among the order embedding variants, the best results are obtained with one hidden layer of 1000 dimensions, which is in accordance with the idea that higher-dimensional embeddings should perform better (cf. Section 3). Based on the validation set, for the order embedding models, the hyperparameter Δ was set to 1 while λ was set to 0.5. Finally, the different variants of the joint premise encoder consistently underperform the order embeddings model. This can most clearly be seen on LP, where these models fail completely. The inductive bias that comes from min-pooling the embeddings of the individual premises thus clearly helps to learn better embeddings.

Impact of the Vocabulary Size. We now explore the effect of two factors: the number of atoms ℓ in a problem instance and the total number of atoms $|At|$ in the vocabulary. Figure 1 compares the order embeddings model with the joint encoder when the number of atoms ℓ is varied while the size of the vocabulary is fixed at $At = 500$ (top), and when the size of the vocabulary $|At|$ is varied while the number of atoms is fixed at $\ell = 5$ (bottom). In all cases, we vary the number of premises between 1 and 10, as before. The models are trained on 100K SAT instances and tuned on 20K SAT instances, as before. For each choice of ℓ and $|At|$ we create corresponding training, validation and test sets.

Let us first consider the top graph of Figure 1. Somewhat surprisingly, when tested on out-of-distribution examples (i.e. LP and RP), the joint encoder is more sensitive to the number of atoms ℓ than the order embeddings. As a result, for $\ell = 250$ we can see that the order embeddings outperform the joint encoder on the LP and RP test sets. For the SAT test set, the joint encoder performs much better, which is in accordance with the finding from Zhang et al. (2023) that this model heavily relies on shortcuts that are dependent on the data generation process. However, when looking at the bottom graph in Figure 1, we can see that the order embeddings model struggles with increases in the overall size of the vocabulary, while the joint encoder is less affected by such changes. This is in accordance with the fact that a minimum of $n = 2^{|At|}$ dimensions are needed to learn perfectly faithful embeddings (cf. Section 3).

Impact of Training Data. We now analyse whether the performance of the order embeddings model

Language Model	SAT	RP	LP
BERT-base	91.5	93.5	85.3
BERT-large	83.7	83.1	71.9
RoBERTa-base	90.7	92.3	82.2
RoBERTa-large	85.6	85.7	72.9
ALBERT-base	91.1	88.4	78.2
ALBERT-large	87.4	84.2	74.7
ALBERT-xlarge	74.7	81.5	71.2
DistilBERT-base	**92.2**	**94.2**	**88.7**

Table 4: Accuracy (%) Order Embeddings (0L) across different pre-trained language models.

may be improved by making changes to the training data. Our default configuration is the same as for the results in Table 1. As a first alternative, we simply increase the number of training examples from 100K to 200K. Next, we consider a setting where 100K standard training examples are augmented with 30K "easy" examples, which are problem instances with at most $m = 2$ premises. The intuition is that including such examples could make it easier for the model to learn meaningful embeddings. Furthermore, we consider a setting where the 100K standard training examples are augmented with examples corresponding to the inference pattern based strategies that were introduced in Section 5.1: *weakening*, *resolution*, *contradiction*, *tautology* and *modus ponens* for the positive examples, and *strengthening*, *abduction* and *random* for the negative examples. The main reason for introducing the augmented strategies was to gain a better understanding of the reasons for the underperformance of order embeddings. In total, we add 15K positive examples (i.e. 3K for each positive inference pattern) and 15K negative examples (i.e. 5K for each negative inference pattern). We also evaluate a variant where we first train the model on the inference pattern based examples (for 5 epochs) and then continue training on the 100K standard training examples (as normal, with early stopping based on validation loss). Finally, we also evaluate what happens when we train on 100K RP examples or 100K LP examples.

Unsurprisingly, the results in Table 2 show that training on 200K examples leads to better results, although the differences are relatively small. However, for all of the other training sets, the performance of the order embeddings model actually decreases, especially for the training set based on the inference patterns. For the joint encoder, the impact of the augmentation strategies is generally small. Finally, when training on RP or LP, the performance on SAT is poor. This is expected given that SAT examples contain negative literals, which is not the case for RP and LP examples.

Evaluation on Standard Inference Patterns. One of the most surprising results in Table 2 was the fact that adding training examples based on the inference patterns led to a deterioration in performance. We now explore this issue in more detail. Specifically, we randomly generated 10K examples for each inference pattern and tested the performance of different models on each of these cases separately. The models were trained on the same 100K SAT examples that were used for the experiments in Table 1. Table 3 shows the results. Surprisingly, we can see that all models struggle with at least one of the positive inference patterns. For instance, the order embeddings model recognises almost none of the instances of tautology and only 21% of the instances of modus ponens. The joint encoder similarly struggles with contradiction and tautology. In the table, we also show two configurations where the training data for the joint encoder and order embedding models was augmented with training examples based on the inference patterns (as in Table 2). Surprisingly, simply adding such examples to the training data has little effect on order embeddings but helps the joint encoder in recognizing tautology. The iterative training approach (i.e. training on inference patterns followed by training on SAT) performs better in order embeddings, however, although the resulting model still misses half of the resolution instances and around 75% of the modus ponens instances.

The order embeddings model uses min-pooling to aggregate the premises. Consequently, the larger the number of premises, the easier it is to end up in a situation where the model predicts entailment, even for randomly chosen premises. We, therefore, conjecture that a model trained on examples with, on average, a large number of premises may be too cautious when evaluated on examples with fewer premises. In other words, the main reason why the model performs poorly in Table 3 may be because these test instances have fewer premises rather than the fact that the model has failed to capture fundamental inference patterns such as modus ponens. Indeed, our primary model is trained using a larger number of premises, while modus ponens and resolution instances have only two premises. To test this hypothesis, Table 3 shows two configurations where we have increased the threshold λ from 0.5 to 1, which makes the model less cautious. As can be seen, this indeed leads to a larger number of positive inference patterns being recognised, while the performance on the negative inference patterns is largely unaffected.

Comparison of Language Models. We are particularly interested in using encoder-only transformer models such as BERT to implement our proposed encoding strategies. Thus far, we have exclusively relied on $\text{BERT}_{\text{base}}$ for our experiments. Table 4

compares this language model with a number of popular alternatives. For this analysis, we use the order embeddings without any feedforward layers, allowing us to evaluate the different LMs most directly. Surprisingly, we can see that larger models typically perform worse than the corresponding base models. In fact, the best overall performance was obtained by DistillBERT$_{base}$. This suggests that the increased capacity of the larger models makes them prone to overfitting. For joint encoding strategies, newer models are likely to perform better, but this goes beyond the scope of our focus on learning formula embeddings.

7. Conclusions and Future Work

The main focus of this paper was to analyse the potential of order embeddings for modelling natural language descriptions of propositional logic formulas. We found that this approach indeed has several advantages over the more common strategy of using joint encoders for reasoning with LMs. The approach based on order embeddings appears to be more robust to out-of-distribution examples and tends to perform better on inputs involving a large number of atoms. Moreover, the joint encoder is limited in practice by the maximum token length of the LM. In contrast, the order embeddings approach is not affected by this issue since each premise is encoded separately. Our results clearly show that min-pooling the embeddings of individual premises is beneficial.

However, the order embeddings also have limitations, the most important one being that the performance quickly goes down as the vocabulary size increases. A natural solution would be to increase the intrinsic dimensionality of the embeddings, but simply using larger LMs did not seem to be effective. One possibility could be to use ensembling techniques such as boosting, which has already proven useful for boosting bi-encoder models in information retrieval (Lewis et al., 2022). However, our initial experiments with this strategy were not successful. As another avenue for future work, the embedding model itself could also be improved, as highlighted by the disappointing performance on examples associated with the inference patterns.

8. Acknowledgements

This research was conducted using the supercomputing facilities at Cardiff University, managed by Advanced Research Computing at Cardiff (ARCCA) on behalf of the Cardiff Supercomputing Facility, HPC Wales, and Supercomputing Wales (SCW) projects. Nurul Ariyani was supported by the Center for Higher Education Funding, Ministry of Education, Culture, Research, and Technology of Republic Indonesia. Steven Schockaert was supported by the Leverhulme Trust (project RPG-2021-140). Zied Bouraoui was supported by the ANR-22-CE23-0002 ERIANA.

9. Bibliographical References

Ralph Abboud, İsmail İlkan Ceylan, and Thomas Lukasiewicz. 2020. Learning to reason: Leveraging neural networks for approximate DNF counting. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 3097–3104. AAAI Press.

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2357–2367, Minneapolis, Minnesota. Association for Computational Linguistics.

Satwik Bhattamishra, Kabir Ahuja, and Navin Goyal. 2020. On the Ability and Limitations of Transformers to Recognize Formal Languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7096–7116, Online. Association for Computational Linguistics.

Piero A Bonatti, Sabrina Kirrane, Iliana M Petrova, and Luigi Sauro. 2020. Machine understandable policies and gdpr compliance checking. *KI-Künstliche Intelligenz*, 34:303–315.

Johan Bos and Katja Markert. 2005. Recognising textual entailment with logical inference. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 628–635, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

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.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3882–3890. ijcai.org.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Antonia Creswell, Murray Shanahan, and Irina Higgins. 2023. Selection-inference: Exploiting large language models for interpretable logical reasoning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Maxwell Crouse, Ibrahim Abdelaziz, Cristina Cornelio, Veronika Thost, Lingfei Wu, Kenneth D. Forbus, and Achille Fokoue. 2019. Improving graph neural network representations of logical formulae with subgraph pooling. *CoRR*, abs/1911.06904.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Christopher Hahn, Frederik Schmitt, Jens U. Kreber, Markus Norman Rabe, and Bernd Finkbeiner. 2021. Teaching temporal logics to neural networks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Geoffrey Irving, Christian Szegedy, Alexander A. Alemi, Niklas Eén, François Chollet, and Josef Urban. 2016. Deepmath - deep sequence models for premise selection. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2235–2243.

Patrick Lewis, Barlas Oguz, Wenhan Xiong, Fabio Petroni, Scott Yih, and Sebastian Riedel. 2022. Boosted dense retriever. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3102–3117, Seattle, United States. Association for Computational Linguistics.

Sarah M. Loos, Geoffrey Irving, Christian Szegedy, and Cezary Kaliszyk. 2017. Deep network guided proof search. In *LPAR-21, 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Maun, Botswana, May 7-12, 2017*, volume 46 of *EPiC Series in Computing*, pages 85–105. EasyChair.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

David G. Mitchell, Bart Selman, and Hector J. Levesque. 1992. Hard and easy distributions of SAT problems. In *Proceedings of the 10th National Conference on Artificial Intelligence, San Jose, CA, USA, July 12-16, 1992*, pages 459–465. AAAI Press / The MIT Press.

Aditya Paliwal, Sarah M. Loos, Markus N. Rabe, Kshitij Bansal, and Christian Szegedy. 2020. Graph representations for higher-order logic and theorem proving. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 2967–2974. AAAI Press.

Kyle Richardson and Ashish Sabharwal. 2022. Pushing the limits of rule reasoning in transformers through natural language satisfiability. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 11209–11219. AAAI Press.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, Lisbon, Portugal. Association for Computational Linguistics.

Swarnadeep Saha, Sayan Ghosh, Shashank Srivastava, and Mohit Bansal. 2020. PRover: Proof generation for interpretable reasoning over rules. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 122–136, Online. Association for Computational Linguistics.

Steven Schockaert. 2023. Embeddings as epistemic states: Limitations on the use of pooling operators for accumulating knowledge. *International Journal of Approximate Reasoning*, page 108981.

Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L. Dill. 2019. Learning a SAT solver from single-bit supervision. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2021. ProofWriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3621–3634, Online. Association for Computational Linguistics.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2016. Order-embeddings of images and language. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Mingzhe Wang, Yihe Tang, Jian Wang, and Jia Deng. 2017. Premise selection for theorem proving by deep graph embedding. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 2786–2796.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*, 6:287–302.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Honghua Zhang, Liunian Harold Li, Tao Meng, Kai-Wei Chang, and Guy Van den Broeck. 2023. On the paradox of learning to reason from data. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, pages 3365–3373. ijcai.org.

Botao Zhong, Chen Gan, Hanbin Luo, and Xuejiao Xing. 2018. Ontology-based framework for building environmental monitoring and compliance checking under bim environment. *Building and Environment*, 141:127–142.